UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO DE FIN DE GRADO

García Rodríguez, Pablo

ESTUDIO DE LAS ETAPAS DE AUTOMATIZACIÓN DE UN PROCESO INDUSTRIAL Y SUS IMPLICACIONES EN LA GESTIÓN DE LA PRODUCCIÓN

Director del TFG: Delgado Prieto, Miguel

Co- Director del TFG: Fernández Sobrino, Ángel

Convocatoria: Enero, 2020

SUMARIO

SUMARIO TABLAS

SUMARIO FIGURAS

1. Introducción

1.1 Abstract

En este proyecto se ha realizado la simulación de un proceso industrial en el laboratorio de automatización ubicado en el aula Schneider Electric de la Universidad Politécnica de Terrassa (UPC).

Este proceso industrial cuenta con 5 estaciones de trabajo simuladas en las que estaría ubicado un robot u operario y 1 estación donde se introducirían los palets o bandejas vacías. La línea es capaz de fabricar 3 tipos de productos diferentes en el mismo turno de trabajo seleccionándolos en un dashboard. Se ha realizado una base de datos para almacenar datos provenientes del PLC y posteriormente un dashboard para su visualización.

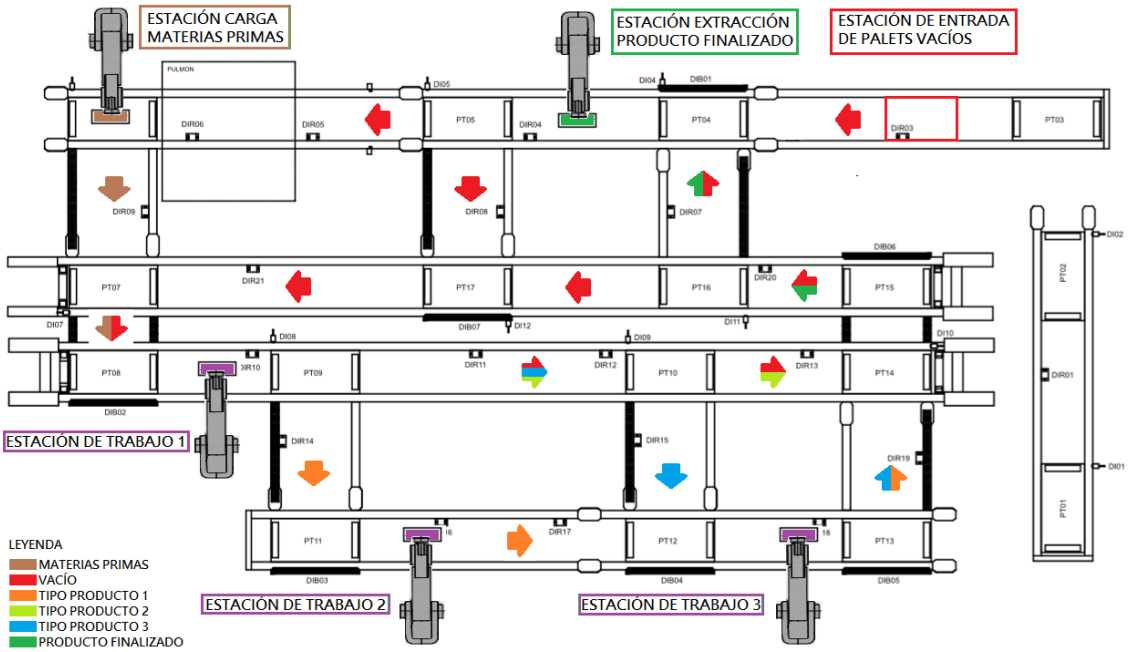


Figura 1 - Flujo de producción de la celda industrial programado

El proyecto consta de la programación del PLC para simular la línea realizada con el software Unity Pro, la utilización del software Node-red como pasarela y Influx DB como base de datos. Por último, el uso del software Grafana para crear una plataforma de visualización de los datos de la línea.

In this project the simulation of an industrial process has been carried out in the automation laboratory located in the Schneider Electric classroom of the *The Universitat Politècnica de Catalunya – BarcelonaTECH* (UPC - Terrassa).

This industrial process has 5 simulated work stations in which a robot or operator would be located and 1 station where empty pallets or trays would be introduced. The line is capable of manufacturing 3 different types of products in the same work shift by selecting them on a dashboard. A database has been created to store data from the PLC and then a dashboard for viewing.

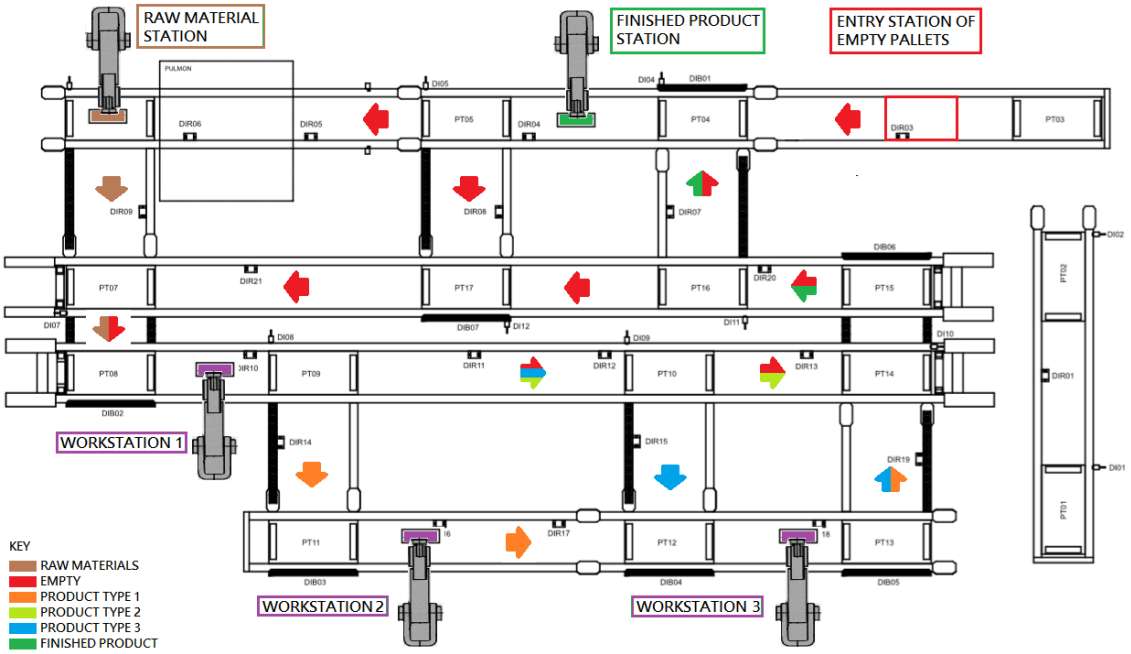


Figura 2 - Cellular layout of production

The project consists of PLC programming to simulate the line made with Unity Pro software, the use of the Node-red software as a gateway and Influx DB as a database. Finally, the use of Grafana software to create a platform for displaying the data of the line.

1.2 Objetivo

El objetivo principal de este proyecto final de grado es el diseño, programación y validación de un automatismo programado sobre un proceso industrial bajo el marco de la Industria 4.0.

La característica principal para el despliegue de la Industria 4.0, es la conectividad y gestión de la información, por lo que se requiere de especial atención a las comunicaciones, el almacenaje de datos y la visualización en remoto de estos para aplicaciones posteriores.

Este objetivo, se llevará a cabo a través de la realización de varios sub-objetivos relacionados con: (i) el diseño y programación de la estructura de comunicaciones, (ii) la puesta en marcha de la aplicación, (iii) la configuración y gestión de la base de datos, (iv) el diseño e implementación de la visualización y (v) la validación del sistema.

1.3 Alcance

Para realizar el objetivo marcado es necesario realizar las siguientes etapas con sus tareas correspondientes, que definen el alcance del proyecto:

Programación de la célula industrial

- Conocimiento de los buses de comunicación de los PLCs.
- Definición de la arquitectura de programación y lenguajes a utilizar.

Adquisición y transmisión de datos

- Conocimiento de los buses de comunicación para poder realizar esta etapa.
- Definición de la arquitectura y flujo de información
- Integración entre nivel de campo y base de datos

Procesado de datos

- Definición de la arquitectura de programación.
- Selección de datos según criterios útiles.

Visualización

- Conocimiento de los buses de comunicación con los que he podido realizar esta etapa.
- Definición de la arquitectura del dashboard.
- Selección de datos útiles para la visualización.

1.4 Justificación

La elección de este trabajo, desde el punto de vista personal, se justifica en gran parte por la oportunidad que supone tener a tu disposición una celda industrial en el laboratorio de automatización de la Universidad Politécnica de Catalunya. Esto facilita la familiarización con el funcionamiento de las líneas automatizadas presentes en gran parte de la industria actual. Además con este proyecto puedo poner en práctica los conocimientos adquiridos al cursar el grado universitario en Ingeniería Electrónica Industrial y Automática y ampliar mis competencias relacionadas con la automatización industrial.

Permite programar dispositivos PLCs con muchos dispositivos de entrada y salida, relacionados entre sí con diferentes protocolos de comunicación, aprender a implementar la industria 4.0 utilizando la información proporcionada por los PLCs para la creación de base de datos y plataformas de visualización o la creación de dashboards para enviar datos a los PLCs.

Cada vez hay más empresas que apuestan por la industria 4.0 ya que facilita la actividad laboral de los trabajadores, mejora la trazabilidad de los procesos, el mantenimiento de equipos e instalaciones, la optimización de recursos y al mismo tiempo reduce los costes de producción.

Las comunicaciones y gestión de datos permiten habilitar canales verticales de comunicación interna accesible para diferentes niveles jerárquicos o departamentos de la empresa. Por ejemplo, un manager podría ser capaz de ver el flujo de producción en directo, el estado de la línea (producción, paro, mantenimiento), datos estadísticos relacionados con la línea, etc.

Otro motivo por el que elegí esta temática para este proyecto fue por la experiencia que adquirí en mi anterior trabajo (análisis de rechazo de línea en la fabricación de productos electrónicos) y que me llevó a la conclusión de la importancia de tener controladas en todo momento las líneas de producción y poder buscar información de lo que haya podido ocurrir durante el proceso de fabricación de cada producto gracias a su trazabilidad del proceso.

Recuerdo una frase que me comentó una vez mi superior: “Tenemos que tener controladas en todo momento las líneas de producción y no que éstas nos controlen”.

Esto se puede conseguir con una trazabilidad que aporte la máxima información de los procesos y nos permita anticiparnos a las posibles incidencias o problemas que pudieran producirse durante el proceso productivo facilitando la resolución de los problemas.

La utilización de la industria 4.0 en el proyecto es un factor crucial en el ámbito de la automatización ya que es el futuro de este campo.

2. Consideraciones previas

2.1 Controladores lógicos programables

2.1.1 Definición y principales características

Un controlador lógico programable, conocido por sus siglas en inglés PLC (programmable logic controller), es un ordenador industrial utilizado en la automatización de procesos electromecánicos que es capaz de procesar los datos procedentes de sensores, botones o cualquier otra señal de entrada de una máquina o línea con el objetivo de controlar los actuadores de la máquina o línea (señales de salida) como pueden ser pistones, motores, válvulas, etc. Los PLC nos permiten controlar automáticamente un proceso industrial.

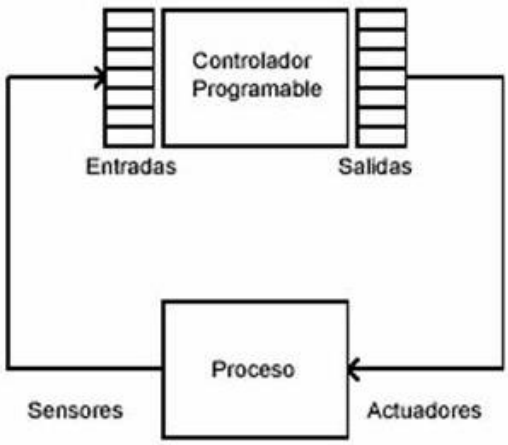


Figura 3- Flujo funcionamiento del PLC

La definición de un PLC de la NEMA (National Electrical Manufacturers Association) es: "Instrumento electrónico, que utiliza memoria programable para guardar instrucciones sobre la implementación de determinadas funciones, como operaciones lógicas, secuencias de acciones, especificaciones temporales, contadores y cálculos para el control mediante módulos de E/S analógicos o digitales sobre diferentes tipos de máquinas y de procesos"

Figura 4 - Imagen ejemplo PLC

Para que un PLC pueda procesar y controlar cualquier sistema (señales de entrada y salidas) es imprescindible que se haya programado previamente la tarea o función que se va a realizar. Para programarlo es necesario un software específico que depende de la marca del PLC. Estos

programas están dotados de diferentes lenguajes de programación que ejecutan secuencialmente instrucción por instrucción.

Los PLC son utilizados en la actualidad en todo tipo de aplicaciones industriales gracias a las siguientes características:

- Un PLC está diseñado para soportar amplios rangos de temperatura que permiten su utilización en diferentes tipos de industrias.
- La inmunidad al ruido eléctrico hace que sea capaz de evitar interferencias.
- La resistencia a la vibración y al impacto presente en los ambientes industriales facilita su utilización.
- La realización de operaciones en tiempo real, debido a su disminuido tiempo de reacción y de ciclo (milisegundos).
- Las posibilidades de programación, la adaptación y utilización de los PLC en diferentes tipos de tarea y/o líneas de producción da lugar a una reducción de los costos que puede suponer la elaboración de nuevos proyectos. Ya que es posible en un elevado porcentaje la utilización de un mismo PLC en diferentes tareas o proyectos una vez se requiere el cambio de este autómata.
- La programación de los PLC se realiza con un lenguaje de programación comprensible para usuarios con nociones medias de.
- Aunque la instalación, programación, etc., debe ser realizada por técnicos cualificados el uso por parte de mano de obra no cualificada es sencillo e intuitivo en los entonos industriales ya que permiten realizar tareas de forma más rápida y evitan la participación de los trabajadores en labores peligrosas.

En la última década ha habido un aumento exponencial de su uso en aplicaciones domésticas o comerciales como el control de la climatización, persianas, puertas, ventanas, etc. También ha aumentado su utilización en aplicaciones de uso general como gestión de iluminación, semáforos, instalaciones de seguridad, etc.

2.1.2 Estructura

Fuente de alimentación

La función de la fuente de alimentación es suministrar la energía eléctrica a la CPU y otras tarjetas del PLC. Los valores más utilizados son ±5V, ±12V y ±24V.

CPU

La unidad central de procesamiento se encarga de interpretar cada una de las instrucciones que el PLC tiene programado en la memoria. Esta unidad consulta los estados de las entradas y elabora las señales de salidas dependiendo de los datos contenidos en la memoria. Actualmente es capaz de realizar miles de instrucciones por segundo.

La CPU está constituida por un microprocesador que puede ejecutar operaciones aritméticas y lógicas que le permiten realizar diferentes funciones. Esta unidad puede testear el PLC con una gran frecuencia para encontrar posibles fallos o errores sucedidos en este dispositivo.

Módulos de entradas /salidas

Estos módulos nos permiten conectar el PLC con el sistema a controlar a través de la coordinación de las señales de entrada, salida e internas del PLC. El módulo de entrada es el encargado de gestionar las señales que le llegan al PLC de la máquina o línea que se quiere controlar. Estas señales provienen de sensores, interruptores, pulsadores, etc. En cambio, el módulo de salida gestiona las señales que provienen de la CPU y que son enviadas a los actuadores del sistema a controlar.

Estas señales pueden ser digitales y analógicas. Por norma general, el número de entradas y/o salidas digitales (8, 16 o 32) es superior a los módulos de señales analógicos (8).

La conexión de los PLC a los dispositivos encargados de generar las señales de entrada o salida se realiza a través de dos tipos de conexiones: Línea GND común (tierra común) y línea VCC común (suministro de potencia común).

Módulo de memorias

La memoria almacena el código de mensajes o instrucciones que tiene que ejecutar la unidad lógica (CPU) del PLC. El tipo de memoria se puede clasificar como:

- Memoria no volátil (memoria de sólo lectura), encargada de almacenar los programas permanentes que coordinan y administran los recursos del equipo y los datos necesarios para ejecutar la operación de un sistema basado en microprocesadores.
- Memoria volátil (memoria de acceso aleatorio), donde se guarda y ejecuta el programa utilizado. Puede ser leída o escrita según la instrucción programada.

2.1.3 Principio de funcionamiento de un PLC

Un PLC funciona cíclicamente, el primer paso que realiza es un trabajo de mantenimiento interno del PLC, como el control de memoria, diagnóstico etc. Seguidamente se actualiza el estado de las entradas leyendo la información de los módulos de las señales de entrada y las convierte en señales binarias o digitales. Estas señales son enviadas a la CPU y se guardan como datos en la memoria. A continuación, la CPU ejecuta secuencialmente el programa creado por el usuario y genera las señales de salidas que dependen del código programado, y las almacena en la memoria volátil del PLC. Por último, las señales de salida generadas son enviadas al módulo de señales para que los actuadores del sistema a controlar puedan leerlas. Cuando se completa un ciclo del PLC, se inicia un nuevo ciclo.

2.2 Protocolos de comunicación

2.2.1 Maestro/esclavo

Maestro/esclavo es el procedimiento para intercambiar información entre varios dispositivos. Un dispositivo (el maestro) controla uno o más dispositivos (los esclavos). El maestro realiza las siguientes funciones: iniciar la comunicación, enviar solicitudes a los esclavos y gestionar la temporización de comunicación. Los esclavos realizan las transferencias de información al responder a las solicitudes procedentes del maestro, no pueden iniciar la comunicación con otros esclavos o el maestro.

El tiempo de respuesta es el tiempo que necesita un esclavo para responder a una solicitud enviada por el maestro.

El maestro dispone de dos modos de comunicación para comunicarse con los nodos esclavos. En el modo de unidifusión, el maestro se dirige a un solo nodo esclavo a partir de una dirección única. El esclavo procesa la solicitud y luego responde al maestro. Las direcciones individuales permitidas se encuentran en el rango 1-247.

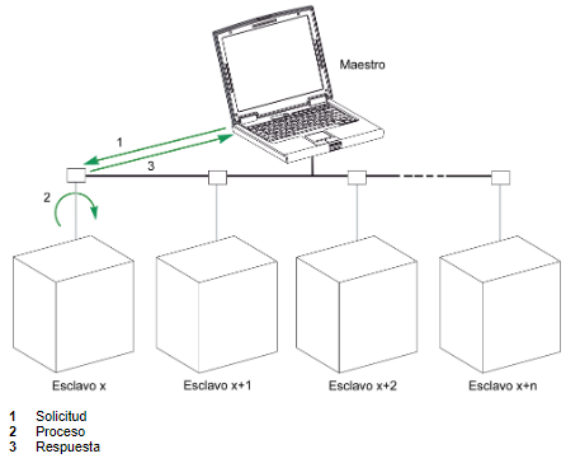


Figura 5 - Modo de comunicación unifusión

En el modo de difusión el maestro envía una solicitud a todos los esclavos utilizando la dirección 0 y estos la aceptan. Los esclavos no responden a los mensajes de difusión.

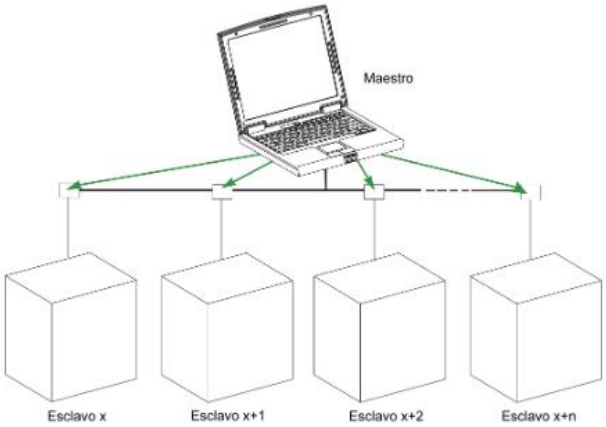


Figura 6 - Modo de comunicación difusión

2.2.2 Cliente/Servidor

Un servidor es una aplicación que ofrece un servicio a un cliente o varios. Una aplicación consta de una parte de servidor y una de un cliente, que se pueden ejecutar en el mismo o en diferentes sistemas.

El cliente se encarga de efectuar una petición o solicitud de servicio que se envía al servidor de la aplicación usando TCP/IP como transporte. Por lo tanto, una petición de datos es iniciada por el elemento que recibirá los datos (el cliente). El envío y recepción de datos con esa estación se realiza cada vez que le toque el turno para comunicarse.

El servidor es un programa que recibe una solicitud, realiza el servicio requerido y devuelve los resultados en forma de respuesta. Generalmente un servidor puede tratar múltiples peticiones (múltiples clientes) al mismo tiempo.

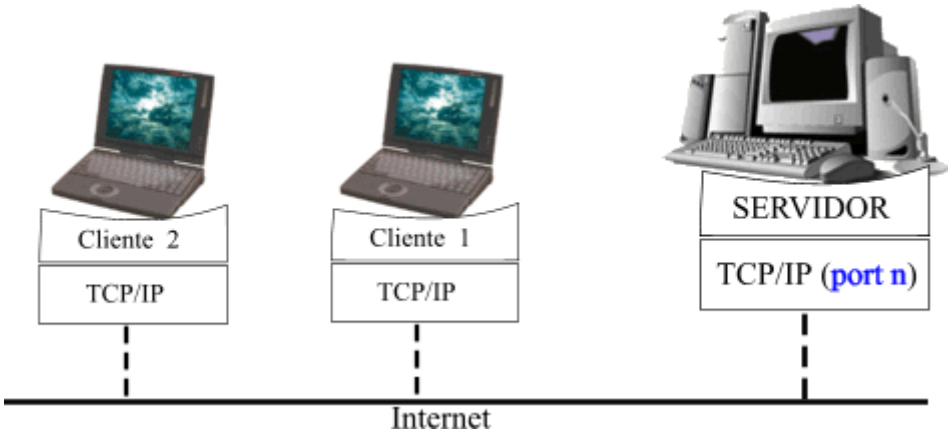


Figura 7 - Comunicación Cliente/servidor

Este modelo de cliente/ servidor se basa en cuatro tipos de mensajes:

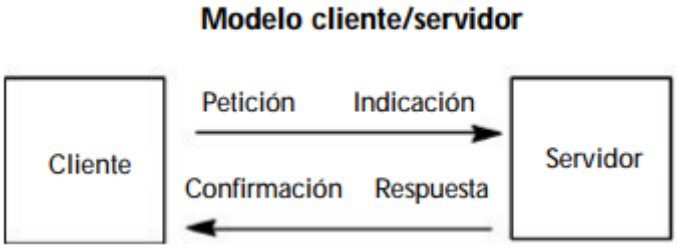


Figura 8 - Flujo de mensajes cliente/servidor

La petición es el mensaje enviado en la red por el cliente para iniciar una transacción. La indicación es el mensaje de la petición recibido por el servidor. Cuando el servidor ha procesado la solicitud envía la respuesta al cliente, la confirmación de MODBUS es el mensaje de respuesta recibido por el cliente.

2.2.3 Producto/consumidor

El patrón de diseño Productor/Consumidor está basado en el patrón Maestro/Esclavo y está orientado a mejorar la forma en que se comparte la información en redes con diferentes velocidades de transmisión. Emplea grupos de buffers en cada estación del sistema de comunicaciones y todas las estaciones, incluido el transmisor, están activos cuando hay actividad en el bus. La transmisión es síncrona.

El Productor es el encargado de producir los datos y los deposita en un buffer de salida. La red copia el contenido del buffer de salida del productor en los buffers de recepción de los consumidores. Estos toman los datos contenidos en los buffers de recepción y consumen los datos producidos por el productor. Esto nos permite almacenar temporalmente los datos, ya que el consumidor puede procesar los datos a su propio ritmo, mientas que al bucle productor puede enviar o recibir datos adicionales al mismo tiempo.

2.2.4 Protocolo Modbus

Es un protocolo de comunicación abierto basado en la arquitectura maestro/esclavo (RTU) o cliente/servidor (TCP/IP). Fue diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLC's) para transmitir datos que no sean críticos o no deterministas entre PLC's. En una red Modbus estándar, hay 1 maestro y hasta 31 esclavos. Modbus admite hasta 249 esclavos y 1 maestro pero el medio de transmisión lo restringe ya que utiliza RS232, RS422 y RS485. El límite máximo para RS232 es 1 maestro y 1 esclavo, y para RS485 y RS422 es un maestro y 31 esclavos pero se puede ampliar usando repetidores hasta el límite de este protocolo.

El tiempo de respuesta es inferior 10 ms para el 90% de los intercambios y el máximo es 700 ms. Cuantos más esclavos tenga la red más lenta serán las comunicaciones.

Los dispositivos monitorizan la red continuamente para detectar el comienzo de una trama y descodifican el campo de la dirección para saber identificar el dispositivo al que va dirigido el mensaje. Cuando el esclavo es el destinatario devuelven un mensaje (modo de unidifusión). En el mensaje de respuesta el PLC añade su dirección en el campo destinado para que el maestro reconozca el dispositivo emisor al descodificar el mensaje. En cambio, no envían respuesta o mensaje cuando las peticiones se envían a todos los dispositivos (modo difusión).

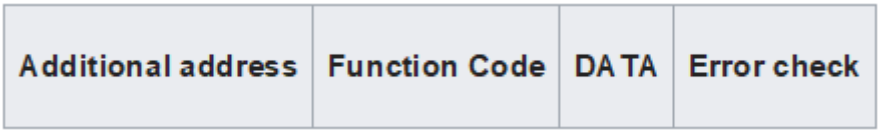


Figura 9 - Estructura trama Modbus

- ***Additional addres (dirección adicional):*** nos indica el destino de la solicitud y su tamaño es un byte. Cuando es 0 es un mensaje enviado en modo difusión como hemos visto anteriormente. Del 1 al 247 el destino es único y de 248-255 está reservado.
- ***Function code (código de función):*** Tiene un tamaño de un byte. Cuando es utilizado por el maestro contiene un código que representa la acción que debe ejecutar el esclavo que utiliza este campo para indicar si la respuesta es normal (libre de errores), enviando la trama original. Cuando se han producido errores envía el mismo código

pero con el bit más significativo con el estado a uno. Ejemplo de una función es 01 ->Read Coils. El maestro está solicitando el estado de sus salidas.

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**		**14**	6.14
			Write File record	**21**		**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Server ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19
			CANopen General Reference	**43**	**13**	**2B**	6.20

Figura 10 - Ejemplos funciones públicas disponibles en Modbus

- ***Data:*** Este campo puede que no exista en algunos mensajes. El maestro introduce información necesaria para que el receptor ejecute la acción determinada por el código de función. Se indica al esclavo en qué dirección debe buscar lo que se ha solicitado a través de la función y también a partir de esa dirección cuantos elementos se deben tomar. Al ser una respuesta por parte de un esclavo contendrá los datos solicitados o un código de error.
- ***Error check (Verificación de errores):*** Tiene un tamaño de dos bytes. Es utilizado para enviar el valor del Checksum que es una función de verificación para detectar cambios accidentales en una secuencia de datos verificando que no haya diferencias después de la transmisión entre los valores obtenidos al hacer una comprobación inicial y otra final.

Hay 4 tipos de datos: las entradas digitales de 1 bit de lectura, las salidas digitales de 1 bit de lectura/escritura, los registros de entrada de 16 bits de lectura y por último los registros de salida de 16 bits de lectura escritura.

Modbus ASCII fue el primer protocolo en serie que generalmente se ejecuta en la capa física RS-232 o RS-485. Cada byte se envía como dos caracteres ASCII y el inicio de la trama se identifica al recibir el carácter ":" (ASCII 3A hex).

Modbus RTU es solo una pequeña variación del protocolo Modbus ASCII. La diferencia está en la codificación de los datos. ASCII codifica el mensaje en caracteres ASCII, mientras que RTU usa bytes, lo que aumenta el rendimiento del protocolo.

2.2.5 Protocolo MODBUS TCP/IP

MODBUS TCP/IP es una variante de la familia de protocolos de comunicación MODBUS para la supervisión y el control de equipos de automatización. Utiliza el protocolo TCP/IP para la comunicación Modbus en un entorno Intranet o Internet conectándose por defecto en el puerto 502. Fue desarrollada en 1999 y proporciona simplicidad, bajo coste y facilidad de desarrollo bajo cualquier sistema operativo.

Una forma sencilla de interpretación del protocolo Modbus TCP / IP es imaginarlo encapsulando un paquete Modbus RTU dentro de un paquete TCP / IP. Modbus TCP / IP utiliza TCP / IP y Ethernet para transportar los datos de la estructura del mensaje Modbus entre dispositivos compatibles. Una red física (Ethernet), con un estándar de red (TCP / IP), y un método estándar de representación de datos Modbus como el protocolo de aplicación).

La función principal de TCP es que todos los paquetes de datos se reciban correctamente y la del IP es asegurar el envío y el destino del mensaje. El TCP/IP no es más que un protocolo de transporte, y no define los datos ni cómo deben interpretarse, esa función corresponde al protocolo Modbus.

El servicio de mensajes de este protocolo utiliza un modelo de comunicación entre dispositivos Cliente/Servidor conectados en una red de Ethernet. Los dispositivos maestros en los protocolos Modbus RTU o ASCII serán clientes en el protocolo TCP/IP. Los dispositivos esclavos serán servidores.

La versión de TCP de Modbus sigue el modelo de referencia OSI. La capa física se estandariza con la norma EIA/TIA 568. En la capa de enlace aparece la dirección física del Equipo (es única para cada ordenador en el mundo). En la capa de Red aparece el protocolo IP que estandariza todo lo relacionado con la direcciones IP. En la capa de Transporte el protocolo utilizado es el TCP y una característica de configuración a resaltar, es la utilización del puerto 502 por defecto.

El formato de la unidad de datos de aplicación es el siguiente, no requiere un Error check ya que las capas inferiores proporcionan protección de checksum, ni un Additional addres porque se utiliza el identificador de unidad.

Figura 11 - Estructura trama Modbus TCP/IP

MBAP (ModBus Application Protocol): Consta de 7 bytes y permite identificar la unidad de datos de aplicación Modbus. Su estructura es la siguiente:

- ***Transaction Identifier (identificador de la transacción):*** Identificación de una petición o respuesta de Modbus (cliente/servidor) (2 bytes).
- ***Protocolo Indentifier (Identificador del protocolo):*** Es igual a 0 en el protocolo Modbus (2 bytes).

- ***Length (longitud)*** tiene un tamaño de 2 bytes y es una cuenta de bytes de los campos identificador de unidad, código de función y los campos de datos.
- ***Unit Identifier (Identificador de unidad):*** Este campo se utiliza para identificar un servidor remoto que se encuentra en una red no TCP / IP (por puente de serie). Nos permite identificar al esclavo (1 bytes).

La Function code y data realiza lo explicado en el protocolo Modbus para estos campos. En los códigos de función y datos con longitud fija, es suficiente con el código de función. En cambio, para los que tienen una cantidad de datos variable, se incluye una cuenta de bytes en el campo de datos.

MODBUS RTU tiene una limitación de 247 nodos por red como se ha comentado anteriormente. Las redes Modbus/TCP pueden tener tantos esclavos como pueda manejar la capa física. Está en torno a 1024, ya que la identificación del PLC de destino se realiza mediante su IP. La rápida adaptación de Ethernet dentro del control de procesos y automatización de la industria ha provocado un rápido crecimiento en los protocolos industriales a través de Ethernet.

2.2.6 Protocolo Profibus

Profibus es un protocolo propiedad de la organización Profibus International. Su nombre procede de PROcess FIeld BUS. A finales de la década de los 90 se consolidó en el mundo industrial gracias al apoyo de compañías como Siemens, Beckhoff, WAGO y Bosch, y se convirtió en la red basada en buses de campos más exitosa del mundo. Es un protocolo abierto ya que permite que los dispositivos de los diversos fabricantes certificados en este bus se comuniquen entre ellos sin necesidad de utilizar interfaces. Está diseñado para la transmisión de datos de E / S de dispositivos de campo, y en el uso de estaciones maestras y esclavas, definidas anteriormente en el apartado Maestro/esclavo.

Profibus se basa en la norma RS485 y puede admitir un máximo de 32 nodos. No obstante, este límite puede ser ampliado hasta 127 nodos mediante el uso de repetidores pero solo pueden ser nodos activos un máximo de 32 y cada repetidor cuenta como un nodo.

El intercambio de mensajes tiene lugar en ciclos donde una estación maestra envía una trama a una estación maestra o esclava que reconoce y envía la respuesta. Esto se produce cuando una estación maestra está en el intervalo de posesión del testigo o token, ya que considera a todas las otras como esclavas al estar en posesión del testigo y tener acceso al bus de comunicación. El paso del token a la siguiente estación maestra se hace incrementando la dirección lógica. Cada estación tiene una lista de estaciones activas y no aceptará el testigo que reciba cuando una estación que no esté identificada en esa lista. Si se produce un segundo intento de la misma estación, en este caso si se aceptará el testigo ya que se asumirá que ha habido un cambio en la configuración de las estaciones maestras y, se marcará a la nueva maestra en la lista como su predecesora.

El tiempo máximo que debería tardar el testigo en volver tiene en cuenta el número de estaciones maestras, la duración de los ciclos de mensajes de alta prioridad y un margen suficiente para los mensajes de baja prioridad y los posibles reintentos.

Los mensajes tienen dos tipos de prioridades: baja o alta. Los primeros mensajes procesados por las estaciones maestras serán los del tipo alto. Una vez iniciado un ciclo de mensaje siempre deberá concluirse, incluyendo los reintentos si son necesarios.

El formato de las tramas asegura una alta integridad se consigue usando bits de start y stop junto con un bit de paridad para cada ocho bits. La sincronización de bit del receptor siempre empieza con el flanco descendente del bit de start.

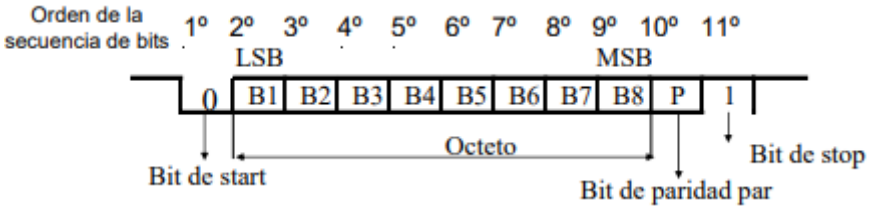


Figura 12 - Estructura trama Profibus

La comunicación puede ser de dos tipos: Broadcast en la que una estación envía un mensaje a todas las estaciones o Multicast donde se envía a un grupo determinado de estaciones.

2.2.7 Protocolo Profibus DP

Es el estándar de Comunicación Profibus utilizado en una parte de línea del proyecto. Sus siglas DP son el acrónimo de Decentralized Periphery. Está diseñada especialmente para la comunicación entre sistemas automáticos de control y E/S distribuidos a nivel de campo.

Las características más destacables son el cambio de más de 1000 Entradas y Salidas con 32 dispositivos en menos de 10 ms (Tiempo de reacción corto), ser un protocolo simple con interfaz de comunicación de bajo coste, excelente diagnosticador al contar con varios diagnósticos en maestro y esclavo, estar dotado de potentes herramientas que reducen el trabajo de configuración y mantenimiento, tener una interfaz de usuario simple.

La mayoría de fabricantes de PLC soportan Profibus.

El ciclo de transmisión de datos está formado por una parte cíclica y otra acíclica. La parte cíclica se realiza en un tiempo fijo el intercambio de datos de las entradas y salidas. La parte acíclica, se realiza en un tiempo variable y también se realizan los Servicios de manejo del bus, inicialización de los esclavos, Funciones de diagnosis y alarmas, lectura y escritura de datos no cíclicos, Comunicación y Repetición de telegramas en caso de fallos.

Profibus permite tener dispositivos con distinto comportamiento y distintos parámetros para cada fabricante con las características documentadas en los manuales técnicos. Para lograr una configuración Plug and Play en Profibus son necesarias las características de comunicación de los dispositivos definidas en un archivo tipo GSD que es proporcionado por el fabricante del dispositivo. Los datos técnicos relativos a la comunicación del archivo GSD, reducen el tiempo consumido en el proyecto de ingeniería, al no tener que buscar esa información en los manuales técnicos.

Los tipos de dispositivos en el protocolo Profibus DP son maestro clase 1, maestro clase 2 y estación esclava.

El maestro clase 1 se caracteriza por ser el controlador central que intercambia datos con los dispositivos de entradas y salidas conectados. Este intercambio de datos se realiza cíclicamente y se ejecutan las tareas usando las siguientes funciones del protocolo:

- ***Set_Prm and Chk_Cfg:*** Para la fase de arranque, transmite parámetros a los esclavos-DP. El número de bytes de datos de E/S con un esclavo DP se define durante la configuración.
- ***Data_Exchange:*** Realiza el intercambio cíclico de datos de E/S con el esclavo DP asignado.
- ***Slave_Diag:*** Lee la información de diagnosis del esclavo DP durante el arranque o durante el intercambio cíclico de datos.
- ***Global_Control***: El maestro DP usa comandos de control para informar a los esclavos DP de los estados de operación. Los comandos de control pueden enviarse a un esclavo individual o a un grupo específico de esclavos DP.

El maestro clase 2 es una herramienta de diagnosis y arranque, normalmente herramienta de configuración. También puede controlar esclavos Los maestros clase 2 soportan funciones para la comunicación con maestros clase 1. Además de las funciones realizadas para el maestro clase 1 soportan las siguientes funciones:

- ***8 RD_Inp and RD_Outp:*** Permite leer datos de E/S de los esclavos DP
- ***Get_Cfg:*** Permite leer los datos de configuración actuales de un esclavo DP
- ***Set_Slave_Add:*** Permite al maestro DP asignara una nueva dirección a un esclavo DP, siempre que el esclavo soporte este método para fijar la dirección

La estación esclava reconoce mensajes o contesta a peticiones. El esclavo sólo intercambia datos de usuario con el maestro cuando el maestro ha cargado los parámetros y la configuración. Un esclavo puede interrumpir la comunicación para procesar información de diagnosis local e interrumpir al proceso en el maestro.

Para la transmisión y acceso al medio hay dos sistemas posibles, monomaestro y multimaestro.

El sistema monomaestro consigue el mínimo tiempo de ciclo de bus. Consiste en un maestro de clase 1 y de 1 a 125 esclavos como máximos. También puede contar con un maestro de clase 2 para la implementación y diagnóstico del sistema.

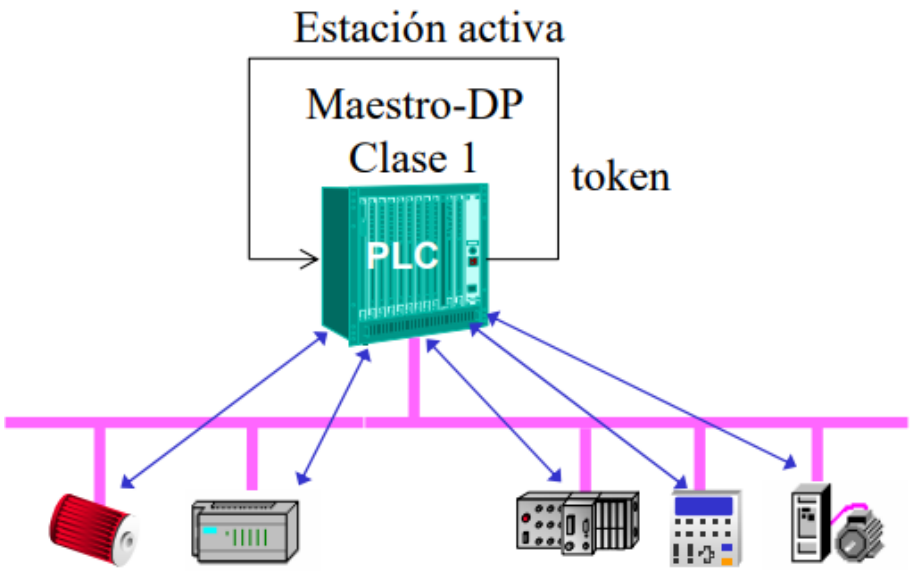


Figura 13 - Sistema monomaestro

En el Sistema Multimaestro varios maestos pueden acceder a un esclavo para leer sus estados. Se componen de múltiplos maestros de clase 1 o 2 y de 1 a 124 esclavos como máximos. Permite un máximo de 126 dispositivos en el mismo bus.

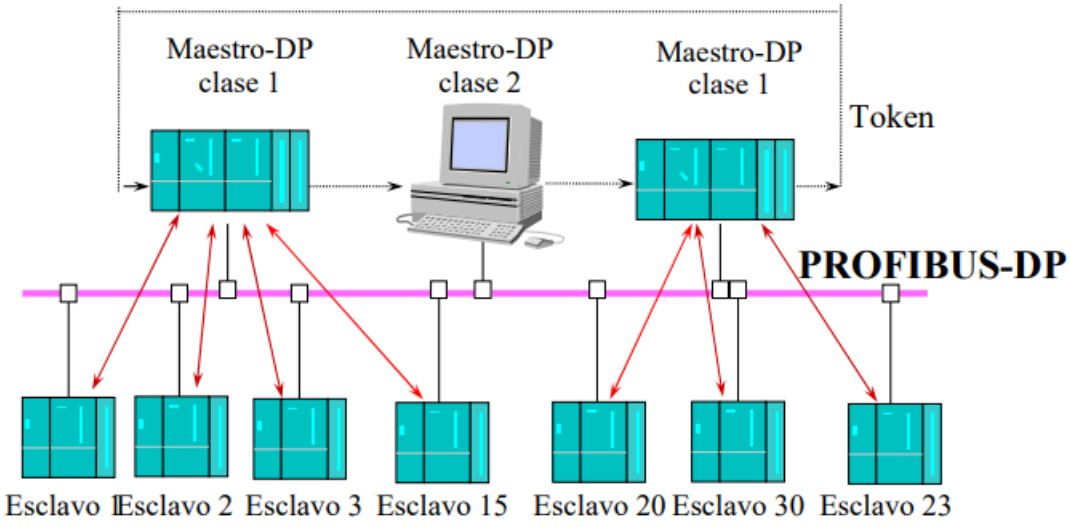


Figura 14 - Sistema multimaestro

2.2.8 Protocolo CAN

CAN es el acrónimo de *Controller Area Network*. Son unidades de mando que están interconectadas. Fue desarrollado por la firma alemana Robert Bosch GmbH en 1986 para reducir la gran cantidad de cableado y conexiones utilizado con el sistema de cableado de Punto a Punto. Con este sistema, una unidad de control, sensor o actuador enviaba la información individualmente a cada receptor a través de cables. Esto suponía un gran peso, complejidad, posibles fallos y costes adicionales. Además, la información no podía ser conocida por otras unidades que no estuviesen conectadas para recibir esta información.

Figura 15 - Distribución unidades de control

CAN es un sistema de bus de comunicaciones serial de alta integridad destinado para comunicar y controlar dispositivos en tiempo real. Tiene una excelente capacidad de detección y aislamiento de errores. La industria del automóvil lo adoptó rápidamente, convirtiéndose en el estándar internacional ISO 11898 en 1993. Desde 1994, se han estandarizado varios protocolos de alto nivel a partir de CAN, como CANopen y DeviceNet , y su uso se ha extendido a otras industrias. Además ofrece una solución a la gestión de la comunicación entre múltiples CPUs (unidades centrales de proceso).

Se transmiten todos los datos (binarios) a través de dos cables, independientemente de la cantidad de unidades de control y de la cantidad de información transmitida. Todas las unidades de control reciben los mismos datos pero solo reaccionan las unidades a las que están destinados los mensajes recibidos.

Las principales características de este protocolo son:

- Capacidad de dar prioridad a los mensajes de la trama.
- Garantía de tiempos de latencia que es el tiempo que tarda en transmitirse un paquete dentro de la red.
- Recepción por multidifusión (multicast) con sincronización de tiempos.
- Sistema multimaestro.
- Detección y señalización de errores. Pudiendo retransmitir automáticamente las tramas erróneas.

- Flexibilidad en la configuración de la red facilitando el número total de nodos posibles (con un máximo de 110 nodos) y su disposición.
- Distinción entre errores temporales y fallos permanentes de los nodos de la red, y desconexión autónoma de nodos defectuosos.

Componentes que integran el CAN-Bus de datos:

- El controlador es el encargado de gestionar el montaje de las tramas CAN, detección de colisiones y/o la comprobación de errores en la transmisión. Este elemento es el que determina la velocidad de transmisión de los mensajes,
- EL transmisor y receptor tienen la misión de codificar y decodificar los mensajes del bus, sincronización y control de los niveles de la señal.
- Resistencias conectadas a los extremos de los cables H y L (High y Low respectivamente), permiten adecuar el funcionamiento del sistema a diferentes longitudes de cables y número de unidades de control del sistema.
- Dos cables por donde se transmite la información del sistema. Tienen las designaciones CAN-High ó H (señales de nivel lógico alto) y CAN-Low ó L (señales de nivel lógico bajo).

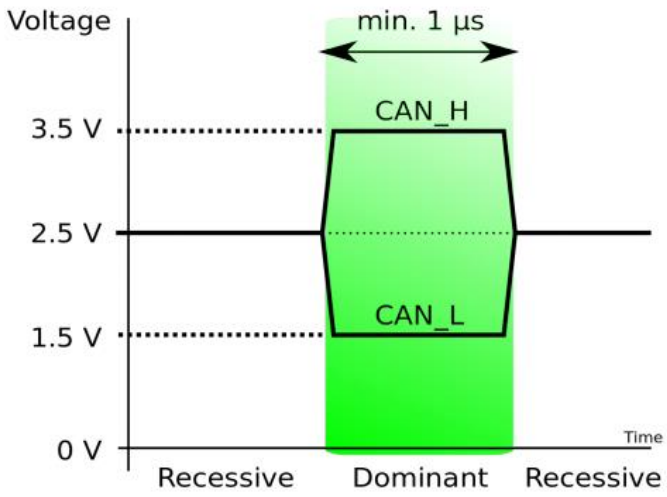


Figura 16 - Ilustración nivel dominante y recesivo

A pesar de que sería suficiente la transmisión de datos con un solo bus de datos, el sistema cuenta con dos buses de datos para optimizar la protección contra los fallos. La información digitalizada es transmitida de forma simultánea en los dos buses de datos pero teniendo signos de tensión inversos. El equipo de autodiagnóstico analiza las dos señales recibidas restando el valor de esas señales. Si el resultado de esta operación matemática no es igual al esperado, el sistema reconoce que alguna de las señales está transmitiendo algún fallo. Estos buses pueden encontrarse en modo recesivo cuando ambos cables tienen el mismo nivel de tensión o en modo dominante cuando hay una diferencia de tensión entre los dos buses CAN_H y CAN_L entre 1,5 y 3 V. Este último modo se utiliza para una mejor protección frente a interferencias

electromagnéticas ya que cuando se realiza la lectura de datos (resta de señales) si han sido sometidas a la misma interferencia la diferencia de voltaje se mantiene constante.

La trama de datos está compuesta por las siete secciones siguientes:

Figura 17 - Estructura trama CAN

- ***Campo de comienzo*** del datagrama marca el comienzo del protocolo de enlace de los datos.
- Los bits del ***campo de estado*** se utiliza como un identificador (único) que permite reconocer la prioridad del mensaje. Cuanto más bajo sea el valor del identificador (más bits con voltaje 0) más alta será es la prioridad determinando el orden de acceso al medio de los mensajes en la línea.
- El ***campo de control*** nos indica la cantidad de bytes del campo datos.
- El ***campo de datos*** transmite la información de la trama.
- ***Campo de aseguramiento*** contiene el código que verifica que los datos fueron transmitidos correctamente mediante la verificación por redundancia cíclica (15 bits) y un bit recesivo (1) que delimita este campo.
- ***Campo de confirmación***. Los receptores informan al transmisor que la recepción del protocolo de enlace de datos se ha realizado correctamente. Cuando se detecta algún fallo, el transmisor recibe inmediatamente un aviso de esa situación. En estos casos se produce la repetición de la transmisión. El campo de confirmación está compuesto por dos bit siempre transmitidos como recesivos (1). Cuando las unidades de mando reciben el mismo campo de confirmación se produce la modificación del primer bit del campo por uno dominante (0), y la unidad de mando que está transmitiendo reconoce que al menos hay alguna unidad de mando que ha recibido correctamente un mensaje. Si esto no es así, la unidad de mando transmisora interpreta que su mensaje presenta un error o en el proceso de envío o recepción se ha producido algún fallo.
- Con el ***campo de fin*** del datagrama finaliza el protocolo de datos. Después de este punto no es posible comunicar un aviso de error, que dé lugar a la repetición de la emisión. Este campo indica la finalización del mensaje con una cadena de 7 bits recesivos.

2.2.9 Protocolo CANopen

CANopen es un protocolo creado hacía el año 1995 que tenía como objetivo establecer una capa de aplicación para el bus de comunicaciones CAN. La capa de aplicación se encarga de acceder a los servicios de las demás capas del modelo OSI y define los protocolos que utilizan las aplicaciones para intercambiar datos. Inicialmente estaba enfocado en la industria del automóvil pero con el tiempo ha evolucionado a un protocolo de comunicaciones industriales multipropósito. Ha sido adoptado como un estándar internacional al estar su uso ampliamente extendido en el mundo industrial.

Fue desarrollado por CiA (CAN in Automotion), asociación sin ánimo de lucro formada por fabricantes y usuarios del bus CAN. Este protocolo proporciona la implementación de un sistema de control distribuido utilizando los servicios de gestión de red y mensajes del protocolo de CAL (CAN Application Layer), también fue desarrollado por CiA.

Esta capa está compuesta por los siguientes servicios:

- ***CMS (CAN-based Message Specification):*** ofrece objetos de tipo variable, evento y dominio. Estos objetos se utilizan en el diseño y especificación para acceder a la funcionalidad de un dispositivo a través de su interfaz CAN. Define 8 niveles de prioridad en sus mensaje
- ***NMT (Network Management):*** facilita los servicios para la gestión de la red. Inicializa, arranca, para o detecta fallos en los nodos. Se basa en el concepto de maestro-esclavo (hay un solo NMT maestro en la red).
- ***DBT (Distributor):*** realiza la asignación de los identificadores CAN compuestos de 11 bits. También se basa en el modelo maestro-esclavo (hay un solo DBT maestro en la red)
- ***LMT (Layer Management):*** permite cambiar ciertos parámetros de las capas. Como por ejemplo el node-id o la velocidad del bus CAN.

El concepto principal añadido por CANopen es el diccionario de objetos (DOD, Device Object Dictionary). Es un grupo de objetos ordenado donde se describe de forma estandarizada la funcionalidad de cada dispositivo y permite su configuración mediante mensajes a través del propio bus. Cada nodo de la red tiene un diccionario de objetos con los parámetros que describen el dispositivo y su comportamiento en la red.

Para garantizar la funcionalidad entre dispositivos de diferentes fabricantes son necesarios una capa de aplicación y unos perfiles

- ***Capa de aplicación (application layer).*** Proporciona un conjunto de servicios y protocolos para los dispositivos de la red.
- ***Perfil de comunicación (communication profile).*** Están descritos todos los parámetros relacionados con las comunicaciones.
- ***Perfiles de dispositivos (device profiles).*** Se definen los objetos de un dispositivo en particular.

Canopen admite la transmisión síncrona y asíncrona de mensajes. Se admiten tres modelos de comunicación: Maestro/Esclavo utilizado en la administración de la red, Cliente/Servidor se

emplea para la parametrización de los dispositivos y Productor/Consumidor se utiliza para transmisión de datos a alta velocidad y que se puede usar para la petición de datos con confirmación de entrega.

2.2.10 Ethernet

Es un estándar de redes de área local con acceso al medio CSMA/CD. Una red local de dispositivos (ordenadores, PLC, etc.) conectados en una zona determinada, no muy amplia como una oficina, el campus de una universidad o incluso una casa particular, etc.

Los dispositivos equipados con Ethernet operan independientemente del resto de los equipos de la red. Estas redes no hacen uso de un dispositivo central de control. Con este estándar se envían los datos agrupados en tramas en un bus de campo compartido que recibirán todos los dispositivos que conforman la red. Cada dispositivo aceptará los datos que lo tienen como destino, no aceptando los datos de la red con destino a otros dispositivos. Todos los dispositivos de una red pueden transmitir información en cualquier momento pudiendo provocar colisiones.

CSMA/CD (Carrier Sense Multiple Access with Collision Detection) hace posible que los dispositivos escuchen la red para determinar si el canal y los recursos se encuentran libres evitando las colisiones.

Preámbulo	Delimitador de inicio de trama	MAC de destino	MAC de origen	Etiqueta	Ethertype	Payload	CRC	Gap

Figura 18 - Estructura trama Ethernet

El Formato de la trama Ethernet es el siguiente:

- El ***preámbulo*** es el primer campo de la trama y marca su inicio, cuando un dispositivo lo recibe, detecta una nueva trama y se sincroniza (7 byte).
- El frame se inicia a partir del ***delimitador de inicio de trama*** (1 byte)
- Las direcciones físicas de los dispositivos de origen y los de destino de los datos están marcadas en los campos de ***MAC*** o dirección de ***destino*** y ***origen*** (6 byte)
- La etiqueta es un campo opcional que indica la pertenencia de la red a una VLAN que son redes lógicas independientes dentro de una misma red física o prioridad (4 byte)
- Cuando se usa un protocolo de capa superior ***Ethernetype*** indica con que protocolo están encapsulados los datos que contiene la Payload, (2 byte)
- La ***payload*** puede almacenar desde un mínimo de 64 Bytes hasta un máximo de 1518 Bytes. Los mensajes con tamaño inferior a 64 bytes indica que son mensajes dañados y/o parcialmente transmitidos
- El valor de verificación ***CRC*** (4 bytes) está incluido en el campo secuencia de comprobación
- El ***gap*** de final de trama se identifica con 12 bytes vacíos para separar las tramas.

Se utiliza cable coaxial, par trenzado o fibra óptica para la transmisión de los datos dependiendo del uso de la red. Cada tipo de cable permite una velocidad de transmisión y una longitud máxima de cable. Hace unos años, Ethernet utilizaba cables coaxiales tradicionales. En la actualidad, los cables de cobre de par trenzado y los cables de fibra óptica son los más

utilizados en el mundo industrial y permiten tasas de transmisión mucho más rápidas y con más alcance.

2.2.11 HTTP

HTTP es el protocolo de comunicación que permite las transferencias de información en la World Wide Web fue desarrollado en 1999 por el World Wide Web Consortium y la Internet Engineering Task Force.

HTTP cuyo significado es Hypertext Transfer Protocol es un protocolo de la capa de aplicación para la transmisión de documentos de hipermedia. Hipermedia es el conjunto de métodos o procedimientos para escribir, diseñar o componer contenidos que integren soportes tales como texto, imagen, vídeo, audio, mapas, etc., con la posibilidad de interactuar con los usuarios.

Fue diseñado para la comunicación entre los navegadores y servidores web utilizando el modelo cliente/servidor y sin estado porque no guarda ningún dato sobre conexiones anteriores. La información relativa a visitas previas se almacena en los cookies ~~en~~ el sistema cliente

El cliente realiza una petición enviando un mensaje, con cierto formato al servidor. El servidor le envía un mensaje de respuesta. Los mensajes HTTP son en texto plano lo que lo hace más legible y fácil de depurar pero por contrapartida son más largos. Tienen la siguiente estructura:

- La ***línea inicial*** en las peticiones está constituida por el método de petición seguido de la url del recurso y la versión HTTP del cliente. Para las respuestas, la versión del HTTP usada seguida por el código de respuesta y la frase de retorno.
- Las ***cabeceras o headers*** permiten al cliente y al servidor enviar información adicional junto a una petición o respuesta. Se pueden clasificar en cuatro grupos:
    - La cabecera general se utiliza en las peticiones o las respuestas pero no tiene relación con los datos que se transmiten en el cuerpo.
    - La de consulta contiene información sobre el contenido que va a obtenerse o sobre el cliente.
    - La de respuesta que contienen información sobre el contenido, como su origen o el servidor.
    - Por último, la de entidad, contienen información sobre el cuerpo de la entidad.
- ***Cuerpo del mensaje***. Es opcional. Su presencia depende de la línea anterior del mensaje y del tipo de recurso al que hace referencia la URL.

HTTP hay una serie predefinida de métodos de petición pero hay flexibilidad para ir añadiendo nuevos métodos y así añadir nuevas funcionalidades.

2.3 Plataforma software de programación

Es un software encargado de la programación y configuración de los autómatas. Permite la configuración del hardware del PLC y la programación del procesador para que ejecute las instrucciones en función del proyecto. Este software permite generar proyectos para PLC de la marca Schneider Electric y de diferentes familias de este fabricante (Modicom M340, Premium, Quantum y Atrium).

Unity Pro XL se basa en la norma, IEC 61131-3. Esta norma define los lenguajes de programación de uso más corriente, las reglas sintácticas y semánticas y el juego de instrucciones fundamentales, etc. Esto nos permite la utilización de un mismo código de programación para diferentes fabricantes y no tener que adaptar el código a cada fabricante al estandarizar la programación de los PLC's.

2.3.1 Lenguajes de programación

Hay cinco tipos de lenguaje utilizados en el software de Schneider Electric. Dos son del tipo literales o escritos y los otros tres son gráficos. Todos estos lenguajes están estandarizados según la norma IEC 61131-3:

ST - Texto Estructurado (Structured Text)

Lenguaje proveniente de ADA, Pascal, y C. Es del tipo literal. Es un lenguaje de alto nivel ya que expresa los algoritmos de una manera adecuada a la capacidad cognitiva humana. Este lenguaje dispone de estructuras para bucles (REPEAT-UNTIL; WHILE-DO), ejecución condicional (IF-THEN-ELSE; CASE), funciones (INC, RE, etc.)

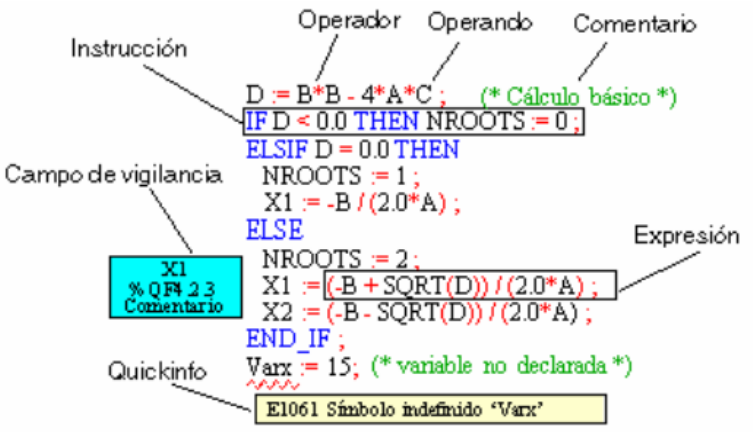


Figura 19 - Ejemplo Texto Estructurado ST

IL.- Lista de Instrucciones (Instruction List)

IL está basado en el lenguaje ensamblador, es de tipo literal y es utilizado en los programas informáticos de bajo nivel. En estos programas las instrucciones ejercen un control directo sobre el hardware y es el lenguaje más parecido al Código utilizado y ejecutado por las máquinas. Funciona a partir de un acumulador simple.

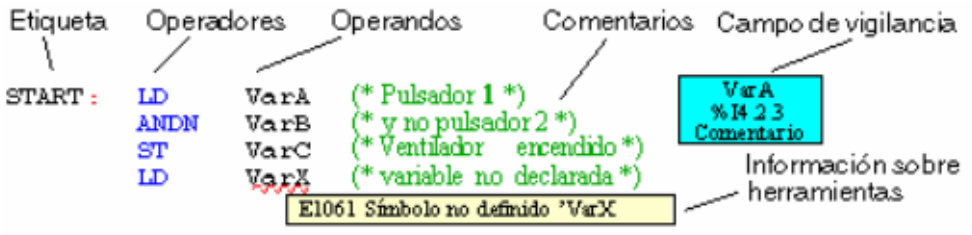


Figura 20 - Ejemplo Lista de Instrucciones IL

LD.- Diagrama de Contacto (Ladder)

Es conocido como Ladder ya que su nombre proviene de la similitud de este código con una escalera (LADDER en inglés) al contar con dos rieles verticales (de alimentación) y "escalones" (líneas horizontales) en los que hay elementos que definen la lógica del circuito mediante funciones.

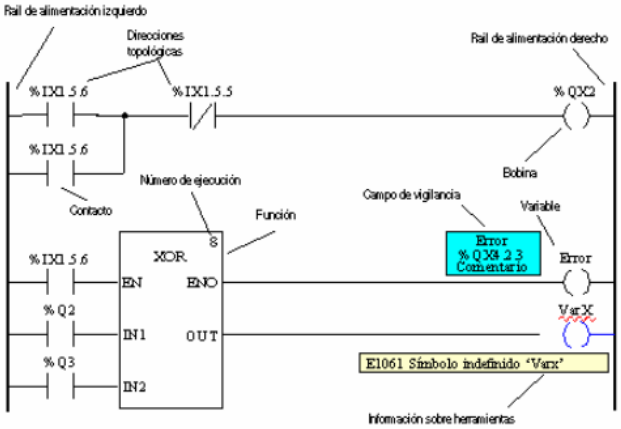


Figura 21 – Ejemplo Diagrama de Contacto (Ladder)

SFC. - Grafcet (Secuencial Funtion Chart)

El GRAFCET es un diagrama funcional que describe los procesos a automatizar, teniendo en cuenta las acciones a realizar y los procesos intermedios que provocan estas acciones. Un GRAFCET está compuesto de las etapas donde se definen los estados del automatismo, la acción asociada que define la función de la etapa y la transición que es la condición o condiciones para cambiar de estado junto con la etapa anterior.

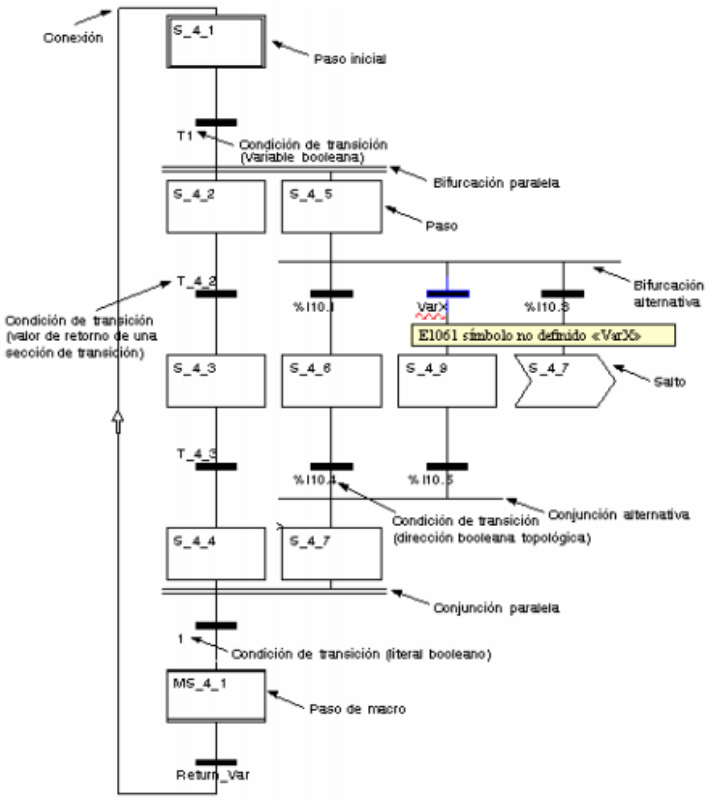


Figura 22 - Ejemplo Grafcet

FBD. - Lenguaje de bloques funcionales (Funtional Block Diagram)

Es un lenguaje de programación basado en las puertas lógicas, como: AND, OR, XOR, etc.

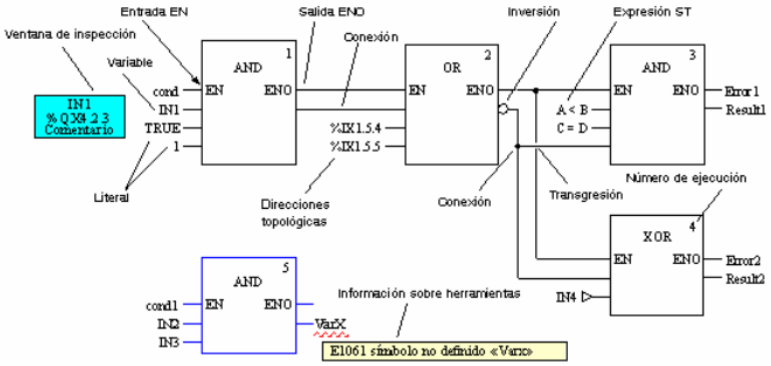


Figura 23 - Ejemplo Lenguaje de bloques funcionales

2.3.2 Secciones

Las secciones son entidades autónomas de programación. Se programan en los diferentes lenguajes (LD, FBD, IL, ST, SFC) que existen en Unity Pro con la condición de que el lenguaje sea admitido en la tarea. Esto nos permite tener varios lenguajes en el mismo proyecto y ordenarlo en diferentes hojas.

La creación de una nueva sección se puede realizar desde el Explorador de Proyectos siguiendo la ruta Programas→Tareas→MAST →Secciones. Desde este apartado clicar el botón derecho y seleccionar Nueva Sección… A continuación, se elige el nombre de la sección y el tipo de lenguaje que se quiere utilizar.

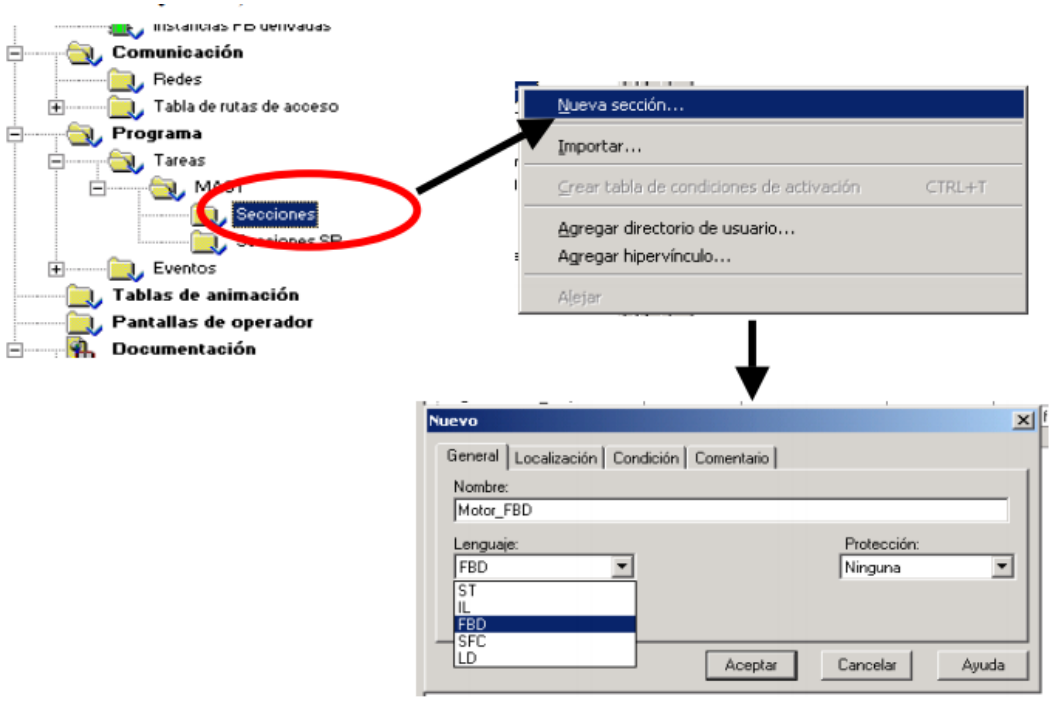


Figura 24 - Crear sección

El orden de ejecución de las secciones seguirá el mismo orden de ordenación que haya marcado en el proyecto. Se puede variar este orden arrastrando las secciones a la posición deseada. Es importante tener en cuenta este factor cuando se programa porque dependiendo del orden de ejecución, un código de programación puede ser eficaz o erróneo.

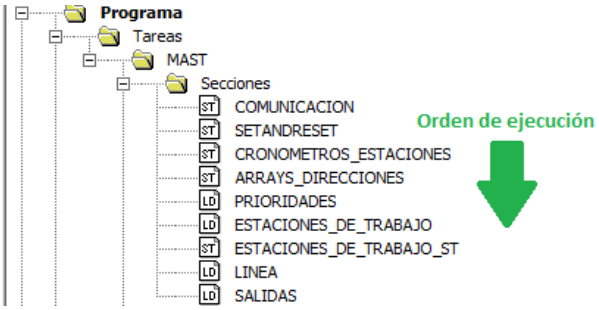


Figura 25 - Orden ejecución de las secciones

2.3.3 Tablas de animación

Una tabla de animación permite ver el valor de las variables, de las direcciones, del estado de los bloques de función. Además, permite modificar o forzar esos valores. Es una gran para comprobar el funcionamiento del código de programación o para solventar posibles errores cometidos.

La creación de una nueva tabla de animación se realiza desde el Explorador de Proyectos y clicando con el botón derecho en Tabla de animación, aparece una ventana donde seleccionaremos Nueva tabla de animación.

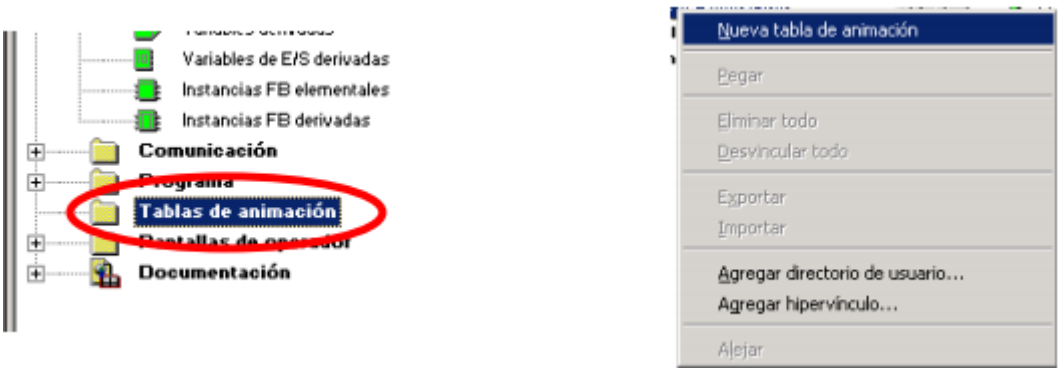


Figura 26 - Crear Tabla de animación

Ejemplo tabla de animación con sus valores en tiempo real:

Figura 27 - Ejemplo tabla de animación

El botón Modificación sirve para modificar solamente el valor de las variables o señales no alocatadas.

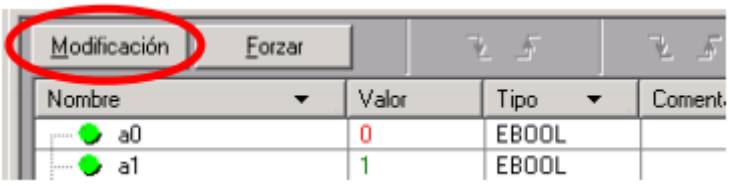


Figura 28 - Modificación variable en tablas de animación

El botón Forzar modifica solamente el valor de las variables o señales alocatadas. Tienen que cumplirse estas condiciones. La variable debe ser de tipo “Ebool”, debe estar localizada y el atributo de forzado debe validarse en el editor de variables.

Figura 29 – Forzar variable en tablas de animación

- Forzar a 1 con el botón: 

- Forzar a 0 con el botón: 

- Cancelar el forzado con el botón: 

2.3.4 Tipos de datos o variables

El software de programación permite crear diversos tipos de datos. En este TFG se explica los utilizados en el código programado de la simulación de producción de la línea industrial.

Una variable es una entidad de Memoria de diferentes tipos cuyos contenidos pueden ser modificados por el programa durante la ejecución. Se clasifican en dos tipos, las variables alocatadas y las no alocatadas.

Una variable alocatada o direccionada está asociada a una referencia de memoria. Por ejemplo en el proyecto, la variable CANTIDAD_M_PRIMAS_ABASTECIDAS se asocia con la palabra de la memoria %MW104.

Figura 30 - Ejemplo variable direccionada

Una variable no alocatada o no direccionada no está asociada a una referencia de memoria (no es posible conocer esta posición en la memoria). En el proyecto no sabemos la posición de memoria que ocupa la variable CONTADOR_BANDEJA.

Figura 31 - Ejemplo variable no direccionada

En el software Unity PRO hay 3 tipos de datos.

EDT (Elementary Data Type):

Son las variables que utilizan un tipo de dato elemental (bool, int, string, etc.).

Elementary Data Type utilizadas:

TIPO	DESCRIPCIÓN
BOOL	Es del tipo booleano. Contiene únicamente el valor FALSE (=0) o TRUE (=1). Ocupa un byte en la memoria, pero el valor se guarda solamente en un bit. El valor predeterminado de este tipo es FALSE (=0).
EBOOL	Es del tipo booleano. Contiene el valor FALSE (=0) o TRUE (=1), pero también incluye información relativa a la gestión de los flancos (ascendentes o descendentes) y el forzado. Ocupa un byte en la memoria. Un bit pertenece al valor (V), otro es el bit de registro (H) para la gestión de los flancos (ascendentes o descendentes) y otro el bit que contiene el estado de forzado (F). 0 sino se fuerza y 1 si se ha forzado.
INT	Es un tipo entero con signo y formato de 16 bits (-32768 hasta 32767)
TIME	El tipo Time T# o TIME# se representa mediante un tipo entero doble sin signo (UDINT). Indica una duración en milisegundos que, aproximadamente, representa una duración máxima de 49 días. Las unidades de tiempo permitidas para representar el valor son días (D), horas (H), minutos (M), segundos (S) y milisegundos (MS).

Tabla 1 - Elementary Data Type

Para crear una variable elemental se siguen estos pasos. Se debe acceder primero al editor de datos en el explorador de proyectos. Hacer doble clic en variables e instancias FB y seguidamente doble clic en variables elementales.

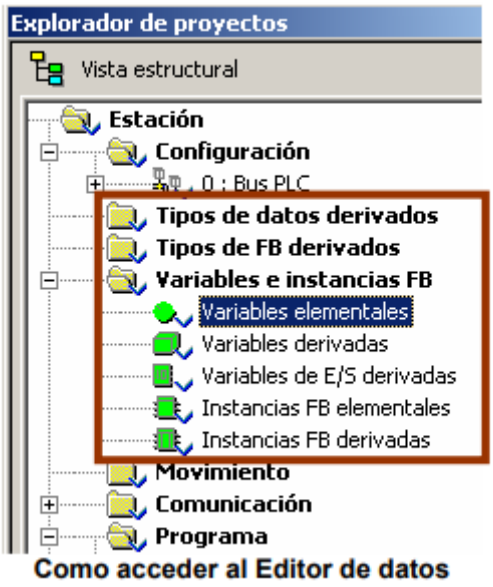


Figura 32 - Ubicación variables elementales

Aparecerán las variables creadas previamente. Al hacer doble clic en la última fila se crea una nueva variable y se tiene que asignar un nombre. Es importante tener en cuenta que el programa no acepta espacios, se pueden substituir por (_). Al finalizar el proceso aparecerá un tipo de dato por defecto que se tendrá que modificar si es necesario. En el campo dirección se debe escribir una dirección física (%I..., %Q..., %IW... %QW...) o de memoria (%M, %MW,...). Al campo valor se le asignará un valor. Hay que tener en cuenta el tipo de dato. Por ejemplo si es un INT aceptará un número entero en cambio sí es un BOOL, aceptará booleanos. Este valor es el que se le asignará a la variable cuando se transfiera el proyecto al PLC o cuando se inicie el PLC. Por último, la columna comentario está habilitada para escribir alguna anotación si se desea.

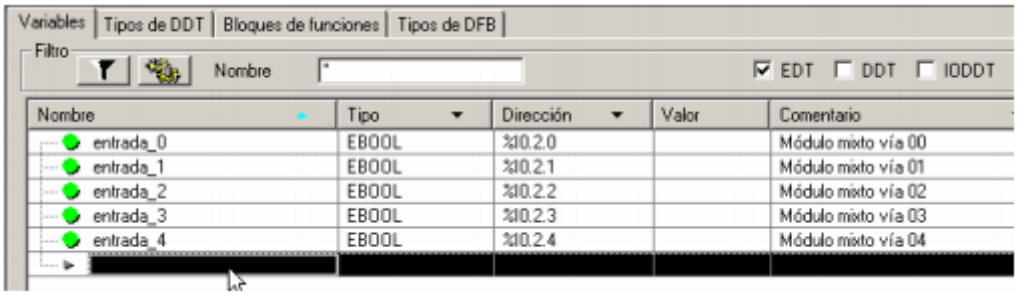


Figura 33 - Crear varaible elemental

Para modificar una variable se debe hacer doble clic en el campo que desea rectificar y validar pulsando la tecla ENTER.

DDT (Derived Data Type)

Son las variables que utilizan un tipo de datos derivado (estructura o arrays).

Para crear un tipo de dato derivado de tipo estructurado, se debe acceder a Variables e instancias FB, seleccionar el comando Abrir del editor de datos y seleccionar la ficha Tipos de DDT.

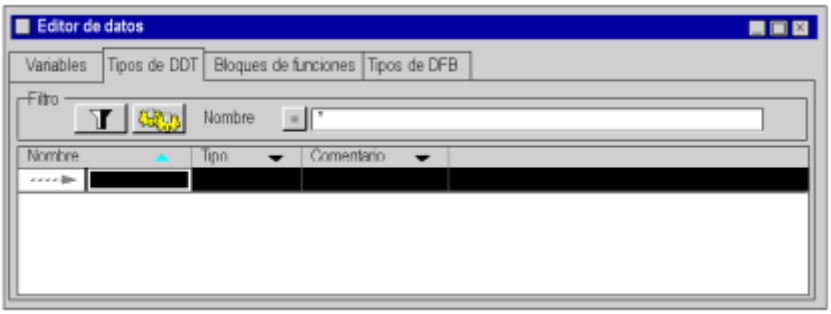


Figura 34 - Crear dato derivado, ficha tipo de DDT

Hacer doble clic en el campo Nombre e introducir el nombre deseado (sin espacios). El tipo predeterminado es <Estruct.>. En el ejemplo es IDENTITY.

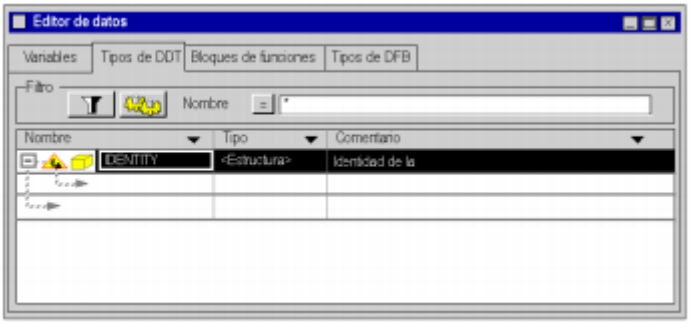


Figura 35 - Crear dato derivado, Estruct

Acceder a la nueva estructura creada clicando +. Para crear el primer elemento de la estructura hacer doble clic en la columna nombre y escribirlo. A continuación seleccionar el tipo de dato. Si se quieren más datos en la estructura hacer doble clic en la celda siguiente. Darle nombre y seleccionar tipo de dato deseado.

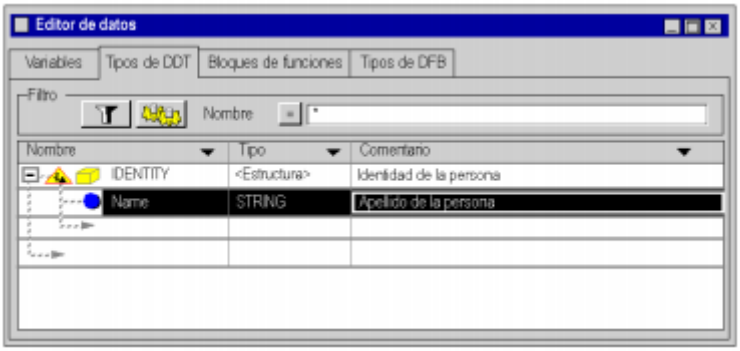


Figura 36 - Crear dato derivado, primer elemento

Una vez creada la estructura hay que analizarla. Se debe situar en el nombre del tipo de la estructura (IDENTITY) y, a continuación, en el menú contextual, seleccionar Analizar tipo. Si se ha realizado correctamente, cambiara el símbolo que aparece delante de la estructura creada.

Figura 37 - Crear dato derivado, analizar tipo

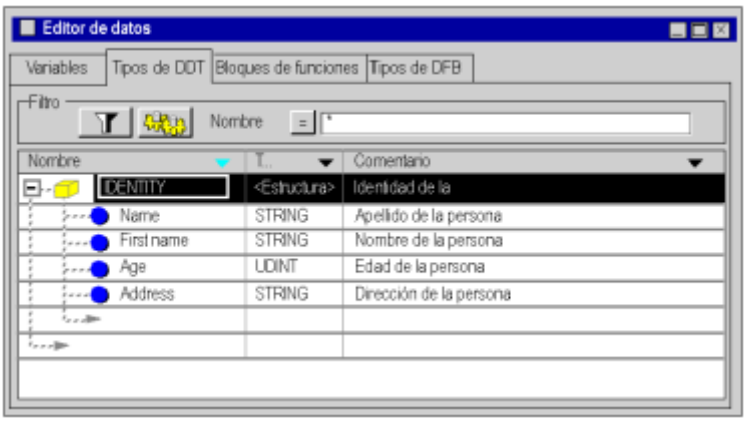


Figura 38 - Ejemplo Estructura

La estructura IDENTITY se ha creado en la vista estructural del proyecto. (Ejemplo)

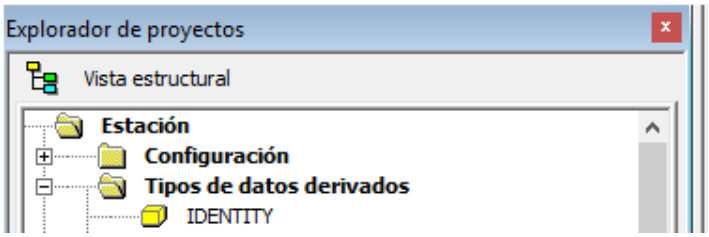


Figura 39 - Ubicación estructura ejemplo

La creación del tipo de dato array se puede realizar en base a dos métodos. El primero se puede realizar en la ficha de Tipos de DDT siguiendo los pasos explicados anteriormente en la creación de la estructura, pero seleccionando el Tipo <Matriz>. A continuación se debe decidir el tamaño de la array y el tipo de dato que habrá en la matriz (pueden ser del tipo STRUCT.). En el ejemplo el tamaño es de 1 … 6 y el tipo de dato es un STRING.

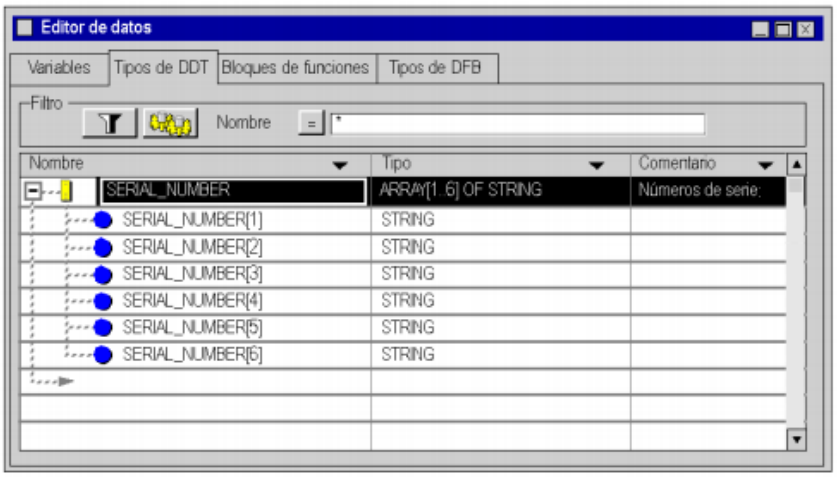


Figura 40 -Crear Array Tipos DDT, Ejemplo

El otro método se crea la matriz accediendo a variables elementales situado en variables e instancias FB. Seguir los pasos explicados en la creación de una variable elemental y escoger el tipo de dato array con su tamaño y tipo de elemento interno. Ejemplo:

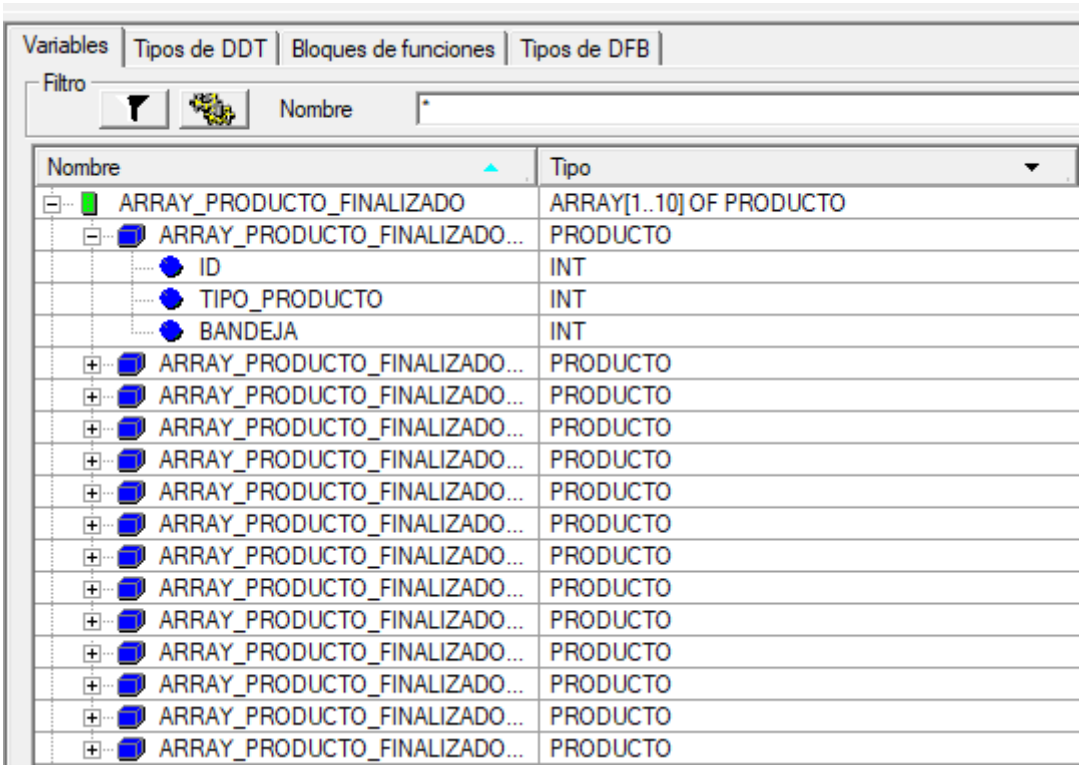


Figura 41 - Ejemplo Array variables elementales

IODDT (Input Output Derived Data Type)

Son las variables de diagnóstico relacionadas con módulos de hardware. Este tipo de variable no se ha utilizado en la elaboración de este trabajo.

2.3.5 Librería de bloques

Es posible insertar bloques de función en los lenguajes de programación LD, ST, FBD y IL. En los lenguajes gráficos (LD y FBD) aparecen en formato de bloque y en formato texto en los lenguajes de tipo de texto (ST y IL). Los bloques de función incluyen un conjunto de librerías y dentro de cada librería se encuentran familias. Esto nos permite tener bloques con funciones ya programadas y utilizar la programación interna con uno de estos bloques ahorrando muchas líneas de programación y agilizando la creación de proyectos. Tipos de bloques:

Función elemental

Las funciones elementales (EF) no disponen de estado interno y sólo cuentan con una salida. Si en las entradas aparecen los mismos valores, siempre que se ejecute la función el valor de la salida será el mismo. Ejemplo: sumatorio de dos valores (el resultado es el mismo).

Bloques de funciones elementales

Los bloques de funciones elementales (EF) tienen estados internos. Si las entradas disponen del mismo valor, el valor de la salida puede variar cada vez que se ejecuten los bloques de funciones. Ejemplo: contador (aumenta el valor de la salida cada vez que se activa el bloque).

Bloques de funciones derivados

Los bloques de funciones derivados (DFB) son bloques creados por el usuario con los lenguajes de programación de Unity Pro (programación FBD, LD, IL o ST).

Los bloques de funciones utilizados en el proyecto se encuentran en el anexo 7.1

2.3.6 Tipos de direcciones

A continuación se exponen cuáles son los tipos de direcciones y la información que pueden gestionar y transmitir.

La nomenclatura de las direcciones se expresa con % seguido de una o dos letras dependiendo del tipo de dirección utilizada. Los tipos de variables están definidas según la norma IEC 61131–3.

Id	Descripción
%I	Entrada digital
%Q	Salida Digital
%IW	Entrada analógica
%QW	Salida analógica
%M	(Memory) bit de memoria para memorizar un estado 0 ó 1
%MW	(Memory Word) palabra de memoria (16 bits) para memorizar un valor de tipo entero 16 bits.
%KW	(Constant Word) palabra constante para definir una constante en el proyecto (no modificable durante la ejecución del PLC)
%S	(System Bit) bit de sistema de configuración del PLC
%SW	(System Word) Palabra de sistema de configuración del PLC

Tabla 2 - Tipos de direcciones

2.3.7 Bits de sistema

Los autómatas Modicon M340, Premium, Atrium y Quantum utilizan bits de sistema ***%Si*** que indican los estados del autómata o que permiten controlar su funcionamiento. Estos bits pueden probarse con el programa del usuario para detectar cualquier evolución de funcionamiento de un procedimiento de procesamiento establecido.

Bit de sistema utilizado en el proyecto es %S13 - 1RSTSCANRUN.

Este bit se activa en el primer ciclo después de la puesta en RUN. Al pasar el PLC de STOP a RUN (incluso después de un arranque en frío con arranque automático en ejecución) se establece el bit de sistema %S13 en 1. Este bit vuelve a ponerse a 0 al final del primer ciclo de la tarea MAST en la modalidad de ejecución. Este bit de sistema está disponible para los PLC's Modicon M340, Premium Atrium y Quantum.

3. Descripción de la celda industrial

3.1 Introducción

La celda industrial nos permite transportar bandejas o palets mediante correas accionadas por motores. Los retenedores se utilizan para detenerlos y las plataformas para cambiar la dirección del flujo del palet. Todos estos elementos están conectados a unos PLCs que permiten gestionar el flujo de palets después de ser programados adecuadamente.

Figura 42 - Imagen celda industrial

La celda industrial también tiene un pulmón que permite almacenar bandejas en su buffer, una pinza para extraer productos y una pinza para coger los palets y trasladarlos a la zona llamada ASI a CAN. Estos elementos no se han utilizado en este proyecto.

Hay 4 tipos de comunicación en los buses de campo de la célula industrial: Profibus, CAN, Ethernet y ASI-INTERFACE. Su distribución en la línea sigue el patrón ilustrado en la siguiente imagen:

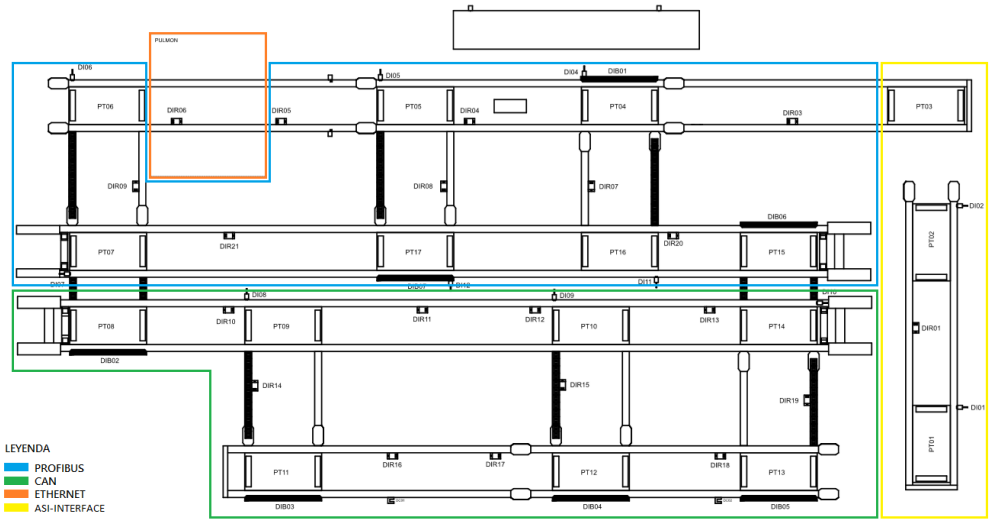


Figura 43 - Distribución protocolos en los buses de campo

Se han utilizado todos los retenedores y plataformas de las líneas Can y Profibus, y el retenedor y el sensor (DIR06) de la estación del pulmón (Ethernet). No se ha utilizado ningún elemento de la línea ASI-INTERFACE.

Cada sección de la célula industrial (Profibus, CAN o Ethernet) utilizada en este proyecto está dotada de islas en la red de bus de campo. Este tipo de distribución permite situar periféricos de entradas y salidas cerca del sistema a controlar y conectarlo con un mínimo de cables al dispositivo maestro (PLC) utilizando el protocolo de comunicación adecuado en cada caso. El PLC lee los datos de las entradas procedentes de los periféricos y realizará su procesado. Posteriormente enviará las señales de salidas a estos periféricos para la activación de los actuadores. Esta distribución permite un ahorro importante en mantenimiento y montaje al disminuir considerablemente el cableado.

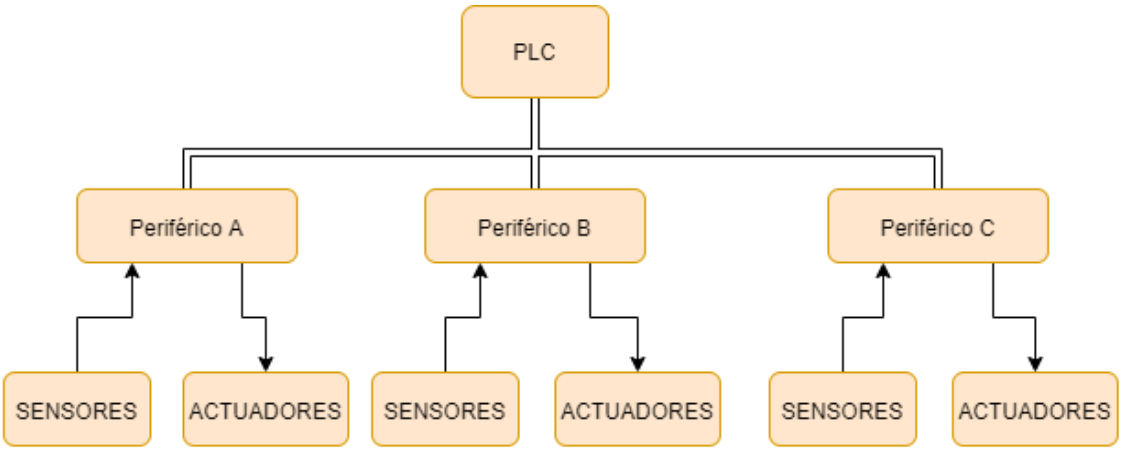


Figura 44 - Flujo Islas Advantys

Estas secciones de la célula (CAN, Profibus, ASI y Ethernet) tienen asociado un PLC con islas Advantys y todos estos PLCs se comunican entre ellos mediante la capa física Ethernet.

3.2 Elementos de la celda industrial

Sensores inductivos

Estos sensores sólo pueden detectar objetos metálicos, pero no objetos no metálicos, como plástico, madera, papel y cerámica. Los sensores inductivos funcionan según los principios de un transformador y utilizan un fenómeno físico que se basa en alternar las corrientes eléctricas.

En la línea industrial hay tres tipos de detectores. Los inductivos que al llegar el palet a la línea detectan una pieza metálica que hay en el lateral cuando la plataforma neumática está bajada.

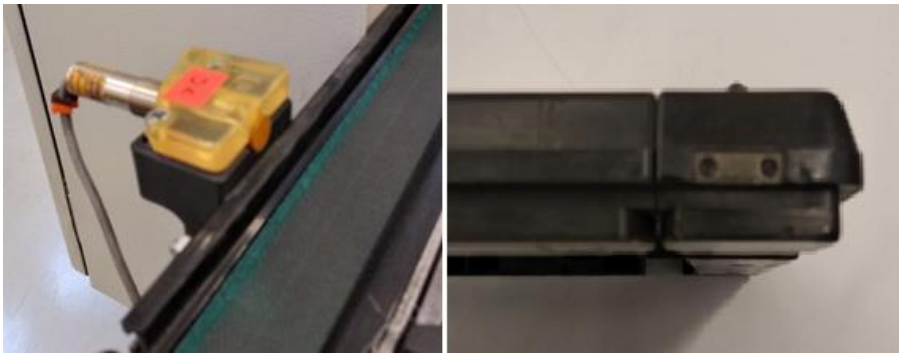


Figura 45 - Sensor inductivo

Los inductores basculantes detectan la llegada o salida del Palet a la plataforma neumática tanto si está subida como bajada. Su funcionamiento se basa en el movimiento de un basculante que activa o desactiva un sensor inductivo.

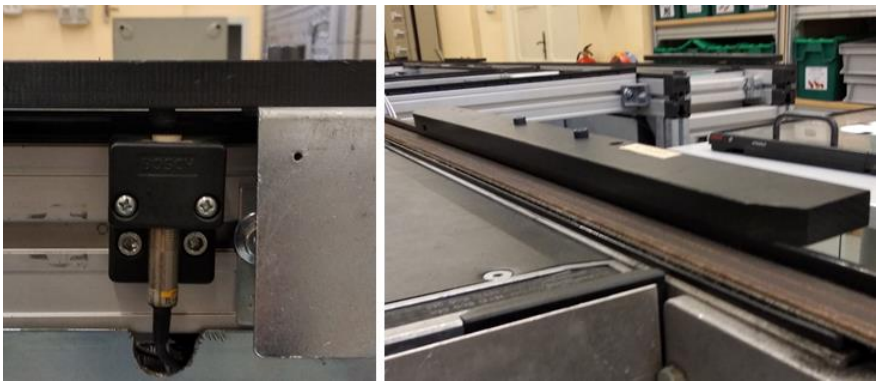


Figura 46 - Sensor inductivo basculante, no detecta llegada Palet

Figura 47 - Sensor inductivo basculante, detecta llegada Palet

Los inductivos con retenedor detectan la llegada del Palet a un retenedor mediante la pieza metálica situada sobre el soporte de plástico del Palet.

Figura 48 - Sensor inductivo retenedor y pieza metálica detectada

Sensores fotoeléctricos de fibra óptica

Un sensor fotoeléctrico es un dispositivo que detecta la presencia de algún objeto mediante luz (visible o no visible). En función de los valores recibidos de esa luz puede activar o desactivar una señal.

Estos detectores tienen la función de detectar la presencia de algún objeto encima del Palet como podrían ser cajas o algún otro producto. Estos detectores tienen una alimentación entre 12V y 24V DC. No han sido utilizados en la simulación de la línea.

Figura 49 - Sensor fotoeléctrico de fibra óptica

Sensores capacitivos

El sensor capacitivo es un interruptor electrónico que trabaja sin contacto. Aprovecha el efecto que tienen los metales y no metales de aumentar la capacidad del sensor cuando se encuentran en un campo eléctrico. La función más lógica de este sensor es la interacción con un operario de la línea, como pedir un Palet, informar de la finalización del producto, etc. En este proceso se ha simulado en el código programado con un temporizador. No se ha utilizado este sensor en el código de programación por la sencillez que supone la simulación del proceso industrial. Pero su introducción en un proceso real no supondría un gran inconveniente a la hora de programar.

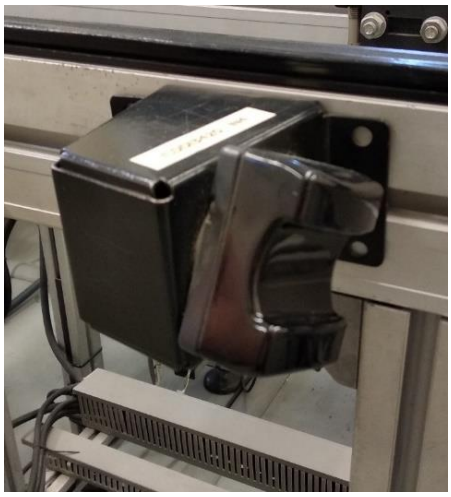


Figura 50 - Sensores capacitivos

Electroválvulas

Las electroválvulas o válvulas solenoides son dispositivos diseñados para controlar el flujo (ON-OFF) de un fluido.

Las electroválvulas accionan los retenedores o plataformas al permitir o denegar el paso de aire comprimido a estos elementos cuando está definido en el programa del PLC realizado por el usuario.

Figura 51 - Electroválvulas

Plataformas

Son los elementos de la línea situados en las intersecciones y permiten el flujo correcto de palets en estas zonas. Hay dos tipos de plataformas según la cantidad de estados que pueden tener. El primer tipo está dotado de dos estados: posición de reposo y subida. El segundo tipo está dotado de 3 estados: posición de reposo, subida y bajada.

El estado de subida permite el flujo de palets entre las zonas con diferentes alturas en la línea.

El estado de bajada permite el flujo sin retención de palets en la plataforma. El funcionamiento es similar a un retenedor. En la posición de reposo retiene el palet y cuando se acciona la bajada permite el paso.

Figura 52- Plataforma en estado de bajada, de reposo y subida

Retenedores

Están formados por un sensor inductivo que detecta la presencia del palet y por un cilindro de simple efecto con retorno de muelle que permite el paso o la retención del palet. Son accionados por electroválvulas.

Figura 53 - Retenedor

Pulsadores

Un pulsador es un interruptor o switch que permite o interrumpe momentáneamente el paso de la corriente eléctrica mientras lo tienes presionado (sin enclavamiento). No se han utilizado estos elementos en la simulación por la facilidad a la hora de realizar la simulación.

Figura 54 - Pulsador

Seta de emergencia

La función de este elemento es la realización de una parada de emergencia en la línea para evitar la aparición de situaciones peligrosas o reducir los riesgos existentes para personas, maquinaria o trabajos en curso. Se tiene que activar solo con la maniobra de una persona.

Figura 55 - Seta de emergencia

Motores cintas transportadoras

Motores que generan el movimiento de las cintas de la línea industrial. Es un motor trifásico de la marca BOSCH conectado en estrella.

Figura 56 - Motor

Contactor

Un contactor es un elemento electromecánico que tiene la capacidad de establecer o interrumpir la corriente eléctrica de una carga, con la posibilidad de ser accionado a distancia mediante la utilización de elementos de comando.

Son los elementos que nos permiten accionar los motores de las cintas transportadoras de la línea a través del PLC.

4. Diseño del sistema de automatización y monitorización

4.1 Introducción

El flujo industrial simulado en la celda industrial del laboratorio de automatización de la Universidad se describe en el siguiente diagrama.

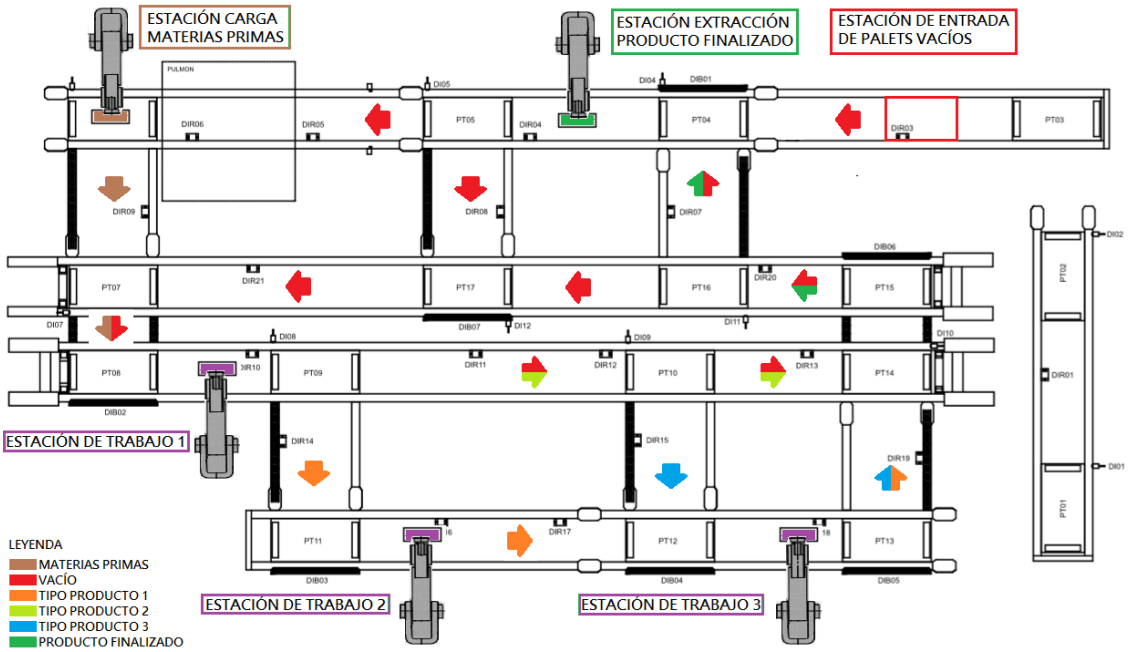


Figura 57 - Flujo de producción de la celda industrial programado

Esta simulación empieza con la entrada de los palets en la estación DIR03 de la celda industrial y se le asigna un ID a los palets. Estos llegan a la intersección PT05 y si alguna de las siguientes estaciones está en reposo DIR05, DIR06 o PT06, el palet seguirá recto. En caso contrario, se desviará en dirección a la plataforma (PT17) y rumbo a la línea Profibus. En esta línea no se realizará ninguna actividad en las estaciones de trabajo y volverá a entrar a la línea CAN por la plataforma PT15. En la plataforma PT16 seguirá recto para introducirse otra vez en la línea Profibus si no hay ninguna de las estaciones DIR05, DIR06 o PT06 libres. Si hay alguna libre se dirigirá hacia la plataforma PT04 y de aquí a la PT05 donde elegirá otra vez la dirección. En todos estos casos el palet está vacío.

La plataforma PT06 realiza el suministro de materias primas y la selección del tipo de producto. En la simulación se realiza mediante un dashboard con el software Node-red.

Una vez cargada la materia prima y escogido el tipo de producto, el palet se dirige a la estación de trabajo 1. En la simulación todos los productos realizan una alguna actividad en esta estación. Al llegar a la intersección de la plataforma PT09, el palet seguirá recto cuando el tipo de producto no sea 1, en este caso se dirigirá a la estación de trabajo 2. Una vez finalice su actividad en esta estación, irá a la línea CAN pasando por las plataformas PT12, 13 y 14. Si el tipo de producto es 2 y 3 llegará a la plataforma PT10 y se decidirá hacia donde se dirige el

palet. Si el producto es 2 seguirá recto y si es 3 se dirigirá a la estación 3 para realizar alguna actividad.

Una vez se han realizado todas las actividades de trabajo en las estaciones, el palet se trasladará y será vaciado en la estación de extracción del producto finalizado (Estación DIR04).

El diagrama de bloques de la celda industrial es el siguiente:

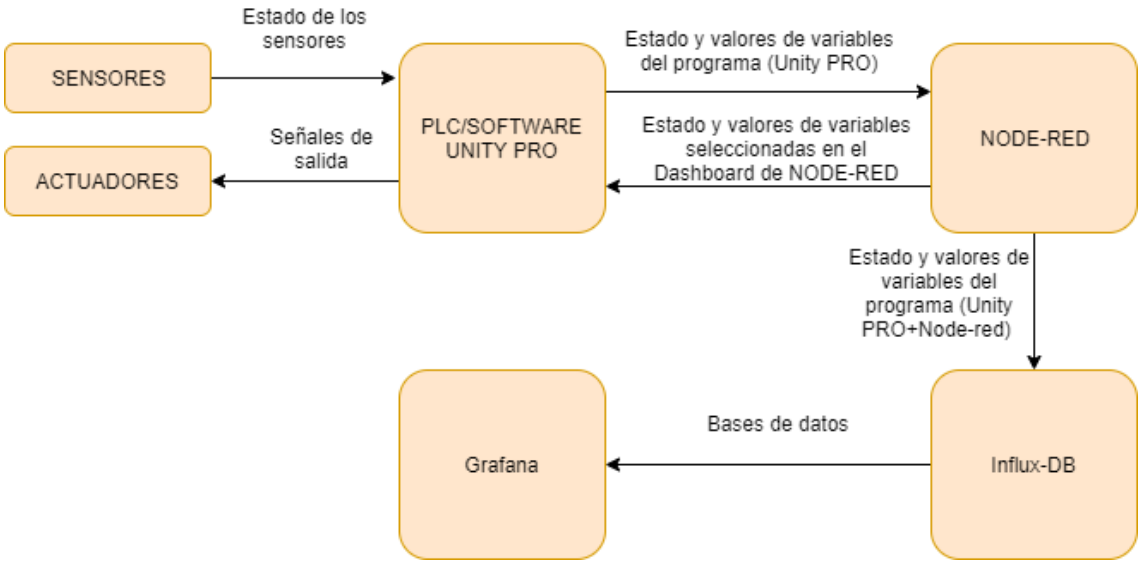


Figura 58 - Diagrama de bloques de la celda industrial

Como ya hemos visto anteriormente, los PLC's necesitan valores de entradas para que una vez compilada la programación pueda enviar señales de salidas. Algunas de estas señales son enviadas (PLC->Node-red) o recibidas (Node-red->PLC) a través del protocolo (MODBUS TCP/IP). En nuestro caso, estas señales de entradas son los estados de los sensores y los valores de las variables determinados en el dashboard de Node-Red. Las señales de salidas del PLC son enviadas a los actuadores para el funcionamiento de la célula industrial pero también se envía información a la pasarela que en nuestro caso es Node-red. Este software una vez procesado esta información envía los datos seleccionados por el usuario o programador al software Influx-DB. Este software se encarga de almacenar estos datos en una base de datos. Por último, la base de datos es leída por el software Grafana que nos permite visualizar estos datos en páginas web de visualización.

4.2 Comunicaciones industriales

Las comunicaciones industriales utilizadas en este proyecto están representadas en el siguiente diagrama de flujo:

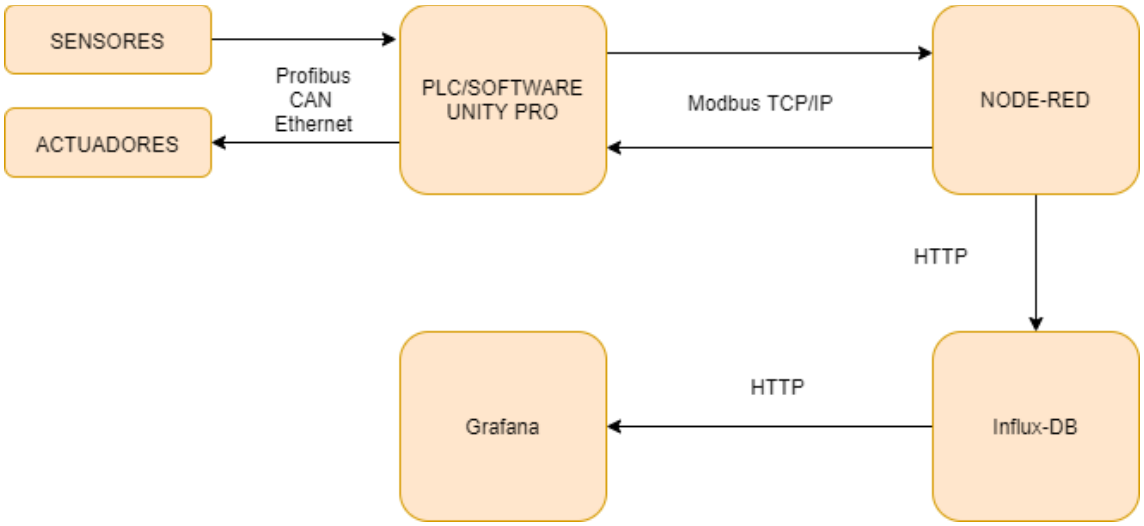


Figura 59 - Diagrama de bloques de la comunicación de la celda industrial

La conexión de las entradas con el PLC y las salidas al PLC se realiza con un puerto serie y con los diferentes protocolos existentes en la célula flexible. El protocolo Profibus (Profibus DP) utiliza la capa física Rs485, CAN (Canopen), utiliza dos cables trenzados con una impedancia característica de 120 Ω y Ethernet utiliza un cableado RJ-45 en esta célula.

La capa física Rs485 a través de un par trenzado permite una velocidad máxima de 10 Mbit/s (a 12 metros) y una longitud máxima de alcance de 1200 metros (a 100 kbit/s).

Los cables trenzados en el protocolo CAN permiten una velocidad máxima de 1 Mbit/s (a 30 metros) y una longitud máxima de alcance de 5000 metros (a 10 kbit/s).

La capa física de Ethernet puede estar formada por cable coaxial voluminoso, par trenzado o fibra óptica. En esta línea se ha utilizado un cableado RJ-45 formado por ocho conexiones eléctricas.

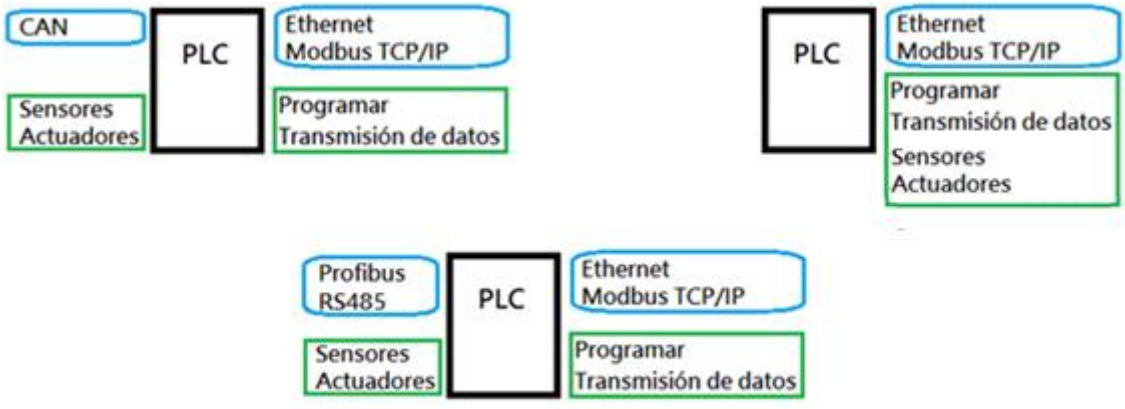


Figura 60 - Protocolos PLCs

En la comunicación entre el PLC y el Node-red se utiliza como aplicación el protocolo Modbus TCP/IP. La capa física utilizada es Ethernet a través de una red de IP y uso TCP como transporte.

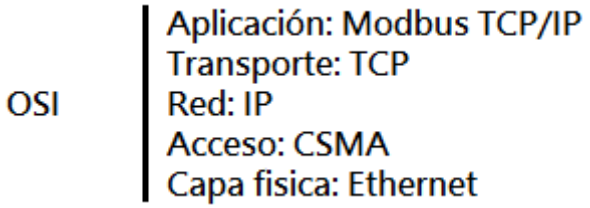


Figura 61 - Modelo OSI - PLC ->Node red

La comunicación entre Node-red y Influx DB se realiza a través de la capa física Ethernet utilizando la aplicación HTTP. La comunicación entre Influx DB y Grafana utiliza la misma capa física y aplicación.

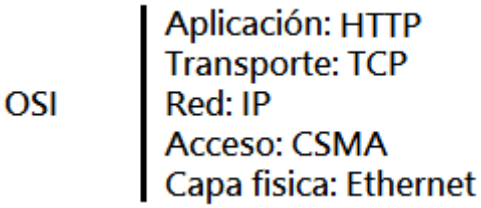


Figura 62 - Modelo OSI - Node red ->Influx DB y Influx DB->Grafana

4.3 Software Node-red

Node-red es una herramienta de visualización open-source, disponible en github, creada en el año 2013 por Nick O'Leary y Dave Conway-Jones del equipo de tecnologías emergentes de IBM (IBM Emerging Technology).

Su objetivo es dar solución a la complejidad que aparece al integrar nuestro hardware con otros servicios permitiendo la interconexión de todos los elementos del IOT (Internet Of Things). Estos elementos pueden ser dispositivos de hardware, APIs o servicios en línea y IOT es la agrupación e interconexión de dispositivos y objetos a través de una red con la capacidad de transferir datos por ella.

La principal característica de Node-red es la sencillez que permite utilizar tecnologías complejas sin tener que profundizar hasta el mínimo detalle en todas ellas. Por ejemplo, gracias a las librerías creadas por la comunidad, se ha podido utilizar fácilmente el protocolo modbus para comunicar con los PLC o unas APIs HTTP para enviar datos a InfluxDB desde Node-red de una manera muy sencilla y fácil.

El editor de flujos de Node-RED consiste en una interfaz basada en un navegador web (HTML) accesible desde cualquier dispositivo. Permite la utilización de nodos que realizan una tarea concreta y conectarlos entre ellos creando flujos de nodos de una forma visual y sin "apenas" tener que programar, ofreciendo un servicio de forma asequible.

El editor se estructura en un entorno gráfico sencillo con:

- Paleta de Nodos: Muestra todos los nodos disponibles en nuestra instalación, con la posibilidad de añadir más nodos desarrollados por otros usuarios e incluso podemos crear e instalar nuestros propios nodos.
- Editor: Nos permite arrastrar nodos desde la paleta y conectarlos. Creado el flujo de operación.

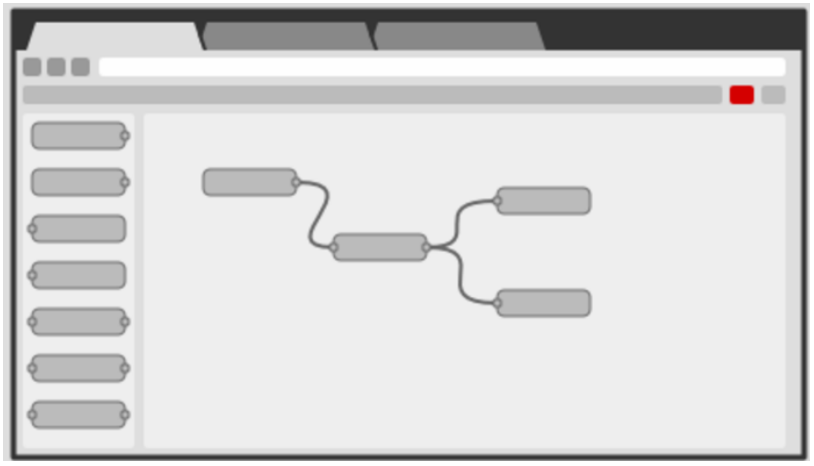


Figura 63- Editor Node red

Esta herramienta visual está programada en NodeJS y puede ejecutarse en una amplia variedad de dispositivos que tienen que ser compatibles con la librería NodeJS o con alguno de los nodos que permiten la comunicación con esta.

Los flujos programados en Node-RED se almacenan internamente en formato JSON y nos permiten utilizarlos en diferentes dispositivos que tengan instalado esta herramienta y los nodos utilizados en el flujo.

En este proyecto Node-red realiza la función de pasarelas que se encargan de traducir la información de los protocolos de nivel inferior utilizados en campo, a protocolos de nivel superior utilizados a nivel de planta.

Con la URL 127.0.0.1:1880 se accede al editor de nodos y con la URL 127.0.0.1:1880/ui se accede al dashboard de Node-red. Los nodos utilizados en el proyectos son:

Function: permite programar un nodo mediante JavaScript.

Figura 64 - Nodo function

Debug: sirve para mostrar mensajes en la zona de debug de Node-RED. Puede habilitarse o deshabilitarse.

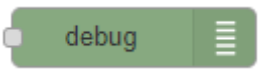


Figura 65 - Nodo debug

Modbustcp: Accede al PLC con el bloque de entrada o salida ModbusTCP/IP. Devuelve los datos en forma de matriz.

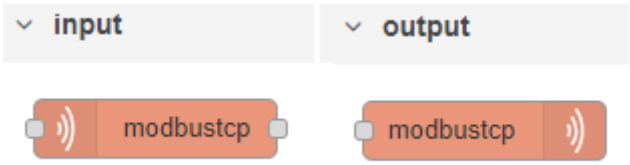


Figura 66 - Nodo Modbustcp

En este proyecto se ha utilizado el código de función FC3: Read Holding Registers o FC6: Write Single Holding Register ya que se quiere leer o escribir posiciones de memoria %MW.

Node input: Se escribe el nombre deseado del nodo, el tipo de lectura que realizará, la dirección de lectura, cantidad de datos de lectura desde la dirección, período entre adquisiciones de datos y elección del servidor.

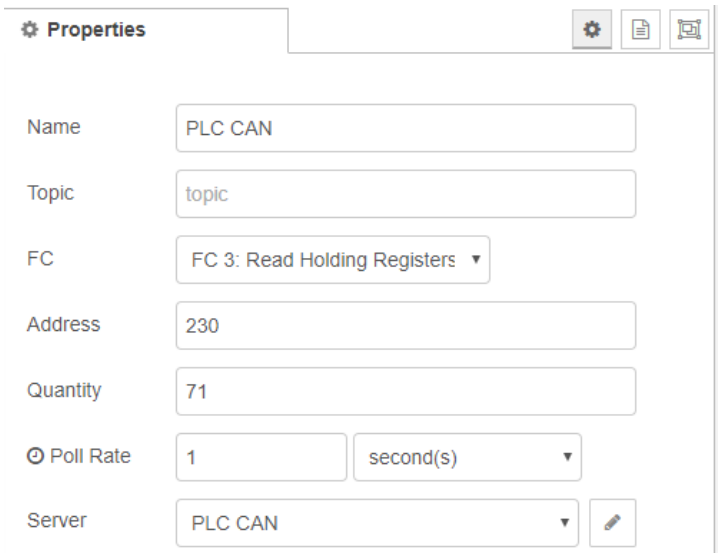


Figura 67 - Propiedades nodo Modbustcp Input

Node output: Se escribe el nombre deseado del nodo, el tipo de escritura que realizará, la dirección de escritura y elección del servidor.

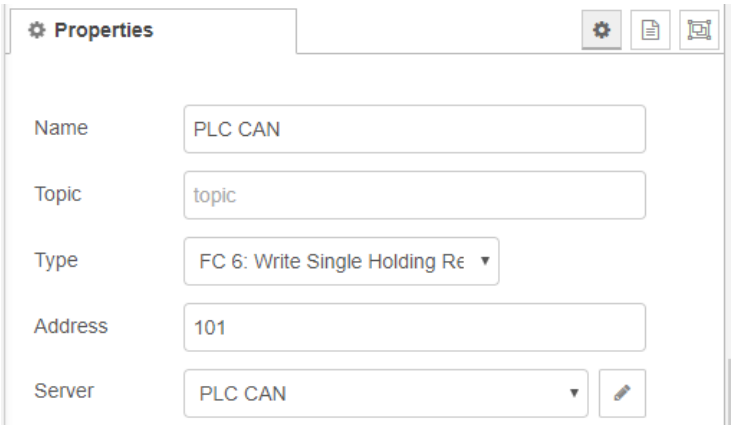


Figura 68 - Propiedades nodo Modbustcp output

En la pestaña server debemos escoger el servidor (PLC). La configuración se realiza clicando en el botón del lápiz apareciendo una ventana de propiedades. Se escribe el nombre del servidor, la dirección IP del PLC, el puerto de comunicaciones 502 que es el puerto por defecto del protocolo Modbus. Ejemplo (PLC CAN).

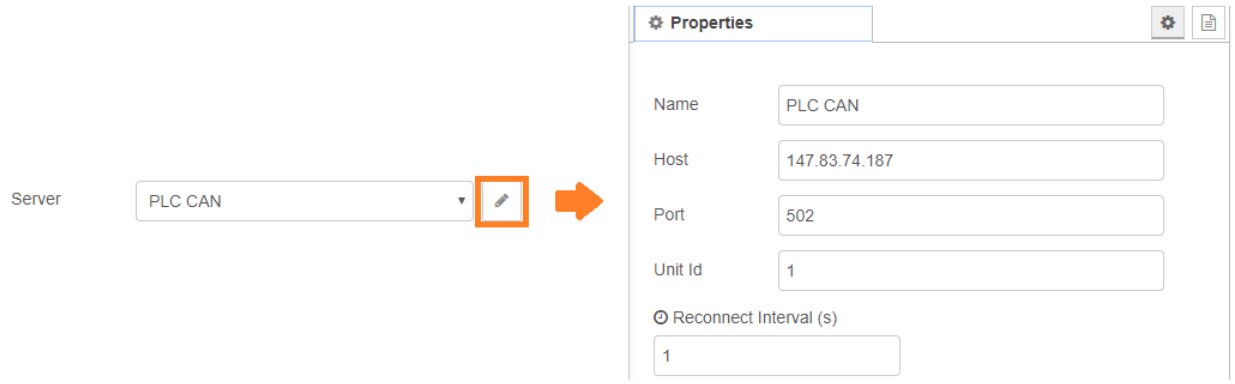


Figura 69 - Nodo Modbustcp Configuración server

Influxd batch: Un nodo de salida influxdb para escribir múltiples fields y tags en múltiples measurements de influxdb.

Figura 70 - Nodo Influx batch

Se selecciona el servidor y para configurarlo se clica el botón del lápiz, se escribe la dirección IP del ordenador (127.0.0.1), el puerto por defecto de InfluxDB es el 8086 y el nombre de la base de datos que se guarda en el ordenador.

En Name se escribe el nombre deseado del nodo.

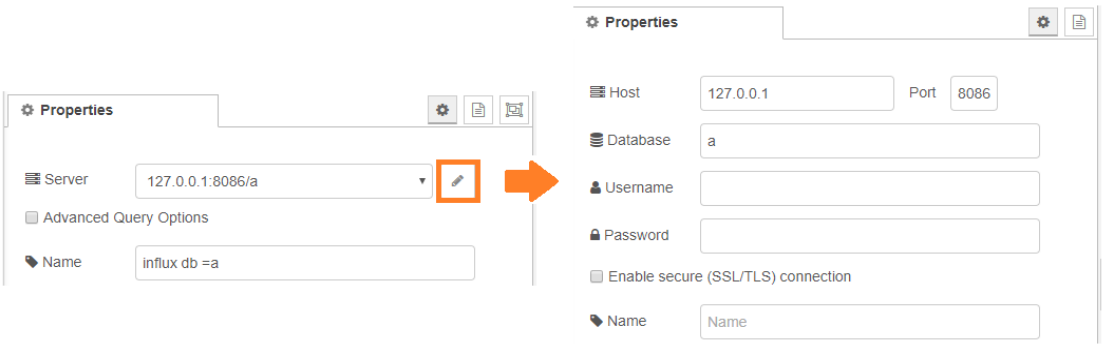


Figura 71 - Nodo Influx batch Configuración server

Button: Agrega un botón a la interfaz de usuario. Al hacer clic en el botón, se genera un mensaje con msg.payload establecido en Payload field.

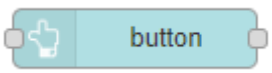


Figura 72 - Nodo button

Numeric: Agrega un widget de entrada numérico a la interfaz de usuario. El usuario puede establecer el valor entre los límites (mínimo y máximo). Cada cambio de valor generará un msg.payload.

Figura 73 - Nodo numeric

Gauge : Agrega un widget de estilo velocímetro a la interfaz de usuario.

Figura 74 - Nodo gauge

4.4 Software InfluxDB

InfluxDB es una base de datos de series de tiempo (Time Series Database) para manejar altas cargas de acciones de escritura y consultas. InfluxDB está especialmente diseñado para almacenar eficientemente una gran cantidad de datos con marca de tiempo, como la supervisión de DevOps, los datos de registro, las métricas de aplicación, los datos de sensor de IoT y los análisis en tiempo real. Permite el procesado y la consulta en tiempo real de estos datos de manera muy eficiente (alto rendimiento).

- Database: es el contenedor lógico que contiene series temporales, usuarios, políticas de retención, etc.
- Measurement: es la estructura en la que se almacenan los datos.
- Timestamp: todo dato almacenado en InfluxDB tiene asociado la fecha y hora. InfluxDB almacena la fecha en formato UTC siguiendo el RFC3339.
- Field: son los valores de datos suministrados almacenados en InfluxDB y asociados a un timestamp. Son datos de tipo strings, floats, integers o booleans. Es obligatorio contar con Fields en nuestra infraestructura de datos.

La instalación de este software es el primer paso para crear una base de datos con InfluxDB. Una vez instalado, se tiene que ejecutar el comando influxd en un terminal apareciendo la siguiente ventana.

Figura 75 - Ventana inicio Influx DB

Cuando se haya realizado este paso se abrirá otro terminal y se ejecutará el comando influx que permite conectarnos y realizar operaciones en la base de datos, como crear base de datos, measurements, visualizar o borrado base de datos, etc.

Comando 'create database' se crea una base de datos (a en el ejemplo)

```
> create database a
```

Figura 76 - Comando create database

Comando 'use' indica la base de datos que se quiere utilizar

```
> use a
Using database a
```

Figura 77 - Comando use

Comando 'select * from 'podemos seleccionar y visualizar el measurement de la base de datos seleccionada (aaa en el ejemlo).

```
> select * from aaa
```

Figura 78 - Comando select from*

Comando 'drop measurement' elimina todos los datos y series del measurement y elimina el measurement del índice.

```
> drop measurement aaa
```

Figura 79 - Comando drop measurement

4.5 Software Grafana

Grafana es un software libre basado en la licencia de Apache 2.0. Fue creado en enero del año 2014 por el suizo Torkel Ödegaard y respaldada por una gran comunidad de entusiasta. Es una herramienta de visualización de series de datos temporales que es muy utilizada para visualizar datos de infraestructuras, aplicaciones o en servicios de monitorización en tiempo real. Permite crear cuadros de mando y gráficas en un dashboard en una página web para poder visualizar los datos obtenidos de diferentes fuentes como las bases de datos Graphite, InfluxDB o OpenTSDB.

Grafana está escrita en Lenguaje Go (creado por Google) y Node.js LTS y con una fuerte Interfaz de Programación de Aplicaciones (API). Su código fuente está publicado en GitHub. Este software ofrece cientos de dashboards y plugins en su biblioteca oficial.

En este proyecto Grafana es el Dashboard con el que se muestra toda la información almacenada en la base de datos de InfluxDB.

Para ejecutar Grafana se abre un nuevo terminal en modo administrador. Una vez abierto el cmd se utilizará el comando 'cd C:\Program Files\grafanalabs\grafana\bin' para acceder al directorio del programa y seguidamente al comando de ejecución 'grafana-server.exe'.

Figura 80 – Grafana Ejecutar como administrador

Cuando se usa el software Grafana se tiene que comprobar que está en ejecución, esto se realiza con el comando 'tasklist' en el cmd y buscando 'grafana-server.exe' para encontrar su PID.

```
Nombre de imagen              PID Nombre de sesión Núm. de ses Uso de memor
========================= ======== ================ =========== ============
grafana-server.exe           4260 Services                   0     26.488 KB
```

Figura 81 - Tasklist

A continuación se ejecuta 'netstat -a –o' en el mismo cmd y se busca el PID anterior para saber el puerto utilizado para acceder al dashboard.

```
Conexiones activas

  Proto  Dirección local          Dirección remota        Estado           PID
  TCP    0.0.0.0:3000           DESKTOP-JQO340A:0      LISTENING       4260
```

Figura 82 - netstat -a –o

En el ejemplo es el puerto 3000 y se accede al dashboard a partir de la siguiente dirección: http://127.0.0.1:3000/

4.6 Retenedores y plataformas

El funcionamiento básico de la línea se basa en la programación realizada con el software Unity Pro de los retenedores y plataformas. Estos dispositivos pueden llegar a tener 5 combinaciones de 3 estados posibles. El estado reposo (REST en el código de programación) se activa (1) cuando no hay ningún palet en el dispositivo. En el estado avance (MOVE) el palet está circulando hasta la siguiente estación. La activación (1) del estado petición (READY) se produce cuando se detecta un palet en el elemento de la línea.

La combinación de estados se visualiza en la siguiente imagen:

Figura 83 - Combinación estados retenedores

C.E.1: Plataforma o retenedor en reposo.

C.E.2: Plataforma o retenedor detecta palet.

C.E.3: Activa el retenedor o plataforma provocando el movimiento del palet.

C.E.4: Palet en movimiento y estado plataforma/sensor en reposo. Se produce en los casos en los que la estación está programada para que cuando esté activado el estado avance, se active el estado reposo cuando el sensor deje de detectar la presencia del palet.

C.E.5: La programación de la combinación de estados 4 permite la entrada de otro palet a la estación del dispositivo activando el estado petición cuando el sensor detecta este elemento, pero su movimiento no es posible porque hay otro palet en movimiento en la línea con destino a otra estación.

El código de programación de estos elementos está en las secciones LINEA y SALIDAS. En Línea está programado el cambio de estados de las estaciones y en Salidas, la activación de las señales de salidas de los retenedores y plataformas.

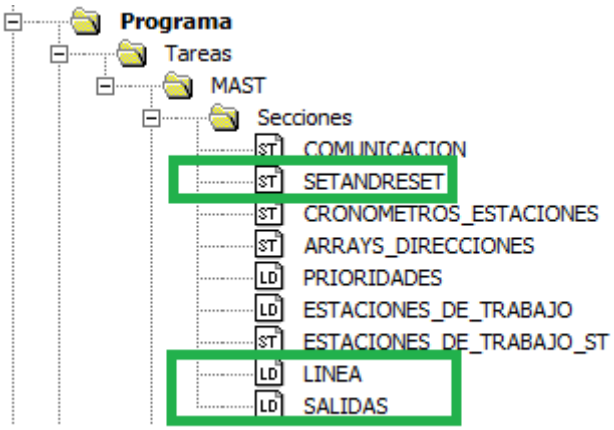


Figura 84 - Secciones programación retenedores

En la sección SETANDRESET, una de las funciones es inicializar los estados de las estaciones teniendo en cuenta que no hay ningún palet en la línea. Por lo tanto, todos los estados de reposo se inicializan con TRUE (activados) y los estados petición y avance se inicializan con FALSE (desactivados) mediante el bit de sistema %S13.

```
DIR04_REST := TRUE;
DIR04_READY := FALSE;
DIR04_MOVE := FALSE;
```

Figura 85 - Estados retenedor SETANDRESET

La activación del estado petición y desactivación estado reposo se produce cuando la estación está en estado reposo y el sensor detecta Palet.

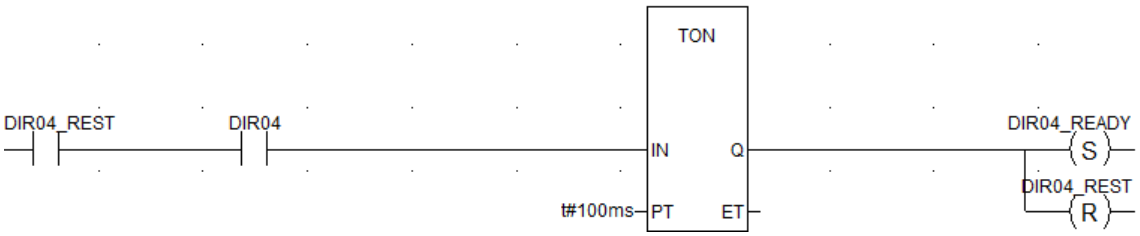


Figura 86 – Línea Ladder activación Ready

El estado del DIR04 está direccionada al sensor físico de la línea que nos informará de su estado (detección o no de Palet). Se realiza el direccionamiento para todos los sensores y sus variables del programa.

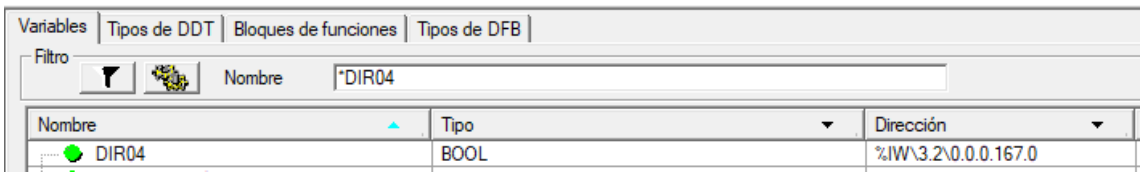


Figura 87 - Direccionamiento sensor retenedor

El estado reposo de la plataforma PT15_REST (%MW303.0) es leído por la línea Profibus de la línea CAN a través de la comunicación entre estas dos líneas ya que necesita saber el estado de esta estación para su correcto funcionamiento. Pasa lo contrario con el estado reposo de la estación PT08_REST (%MW350.0), escrito por la línea Profibus a la línea CAN.

Dirección IP	ID de unidad	Timeout de estado funcional	Velocidad de repetición (ms)	Leer objeto maestro	Leer índice de esclavo	Leer longitud	Último valor (Entrada)	Escribir objeto maestro	Escribir índice de esclavo	Escribir longitud
147.83.74.187	255	1500	60	%MW303	303	1	Establecer en 0	%MW350	350	1

Figura 88 - Comunicación PT15 REST y PT08 REST

SECCIÓN COMUNICACIÓN

PROFIBUS

```
%MW350.0 := PT08_REST;

PT15_REST := %MW303.0;
```

CAN

```
%MW303.0 := PT15_REST;

PT08_REST := %MW350.0;
```

Figura 89 - Sección comunicación PT15 REST y PT 08 REST

La existencia de un fallo esporádico producido por ruido en las señales de los sensores provoca la falsa activación de la petición y seguidamente el estado avance (bandejas fantasmas). Se decide incorporar un temporizador TON para el filtrado de las señales.

La activación estado avance y desactivación estado petición sucede siempre que el estado petición de la estación esté activado, la estación destino esté sin palet (reposo) y no haya ningún palet en movimiento hacia esta estación. La activación del bloqueo de la estación destino nos permite, si es necesario impedir el movimiento de la estación origen.

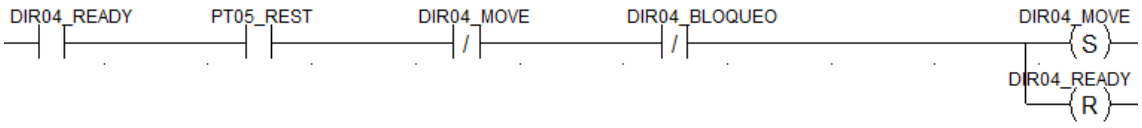


Figura 90 - Línea Ladder activación Move

La programación de la estación cuando un palet entra en una intersección en la que puede desviarse hacia dos destinos, se realiza con el siguiente patrón del esquema Ladder. Es muy parecida al esquema anterior (figura 88) pero con una línea de código para cada destino que active su movimiento. Los elementos comunes son el estado petición y el estado de movimiento negado de la estación. Los elementos de cada bifurcación son el estado reposo de la estación destino, el bloqueo y la dirección. La activación de la dirección nos permite seleccionar la estación destino hacia donde se quiere desplazar el palet, al permitir la activación del avance escogido y el reset del estado petición. Al mismo tiempo, también se desactiva la dirección porque ya ha cumplido su misión. El estado de la variable dirección quedará en espera de la elección del destino del siguiente palet que entre en la estación. Si no se selecciona ninguna dirección, el palet no avanzará al no haberse activado ninguno de los dos movimientos.

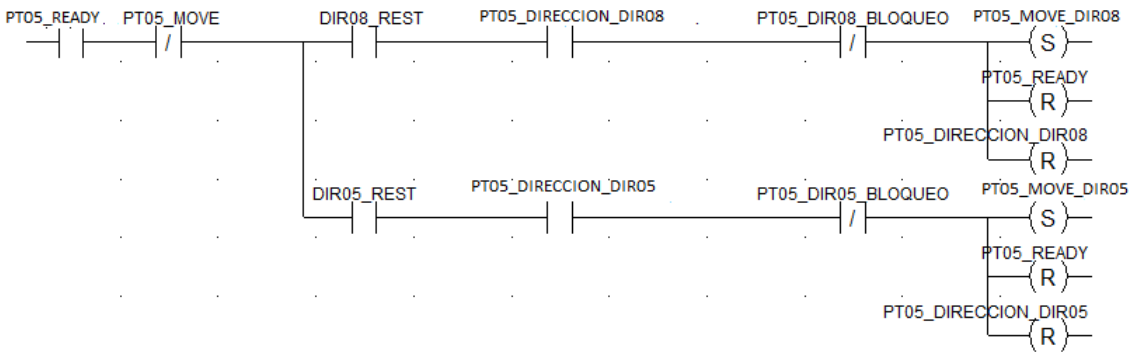


Figura 91 - Líneas Ladder activación Move con direcciones

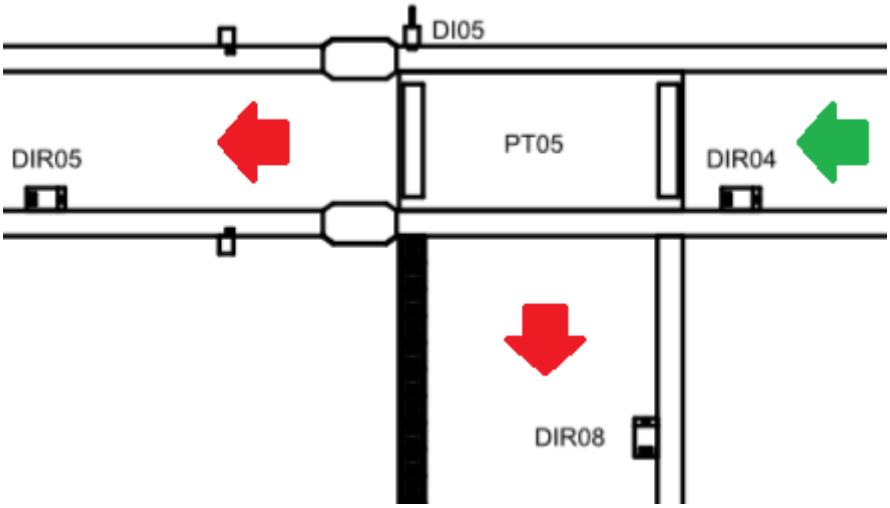


Figura 92 - Flujo plataforma con una entrada y dos salidas

La activación estado reposo y desactivación estado avance sucede cuando el sensor de la estación destino detecta la llegada del palet. Este tipo de programación no permite la combinación de estados 4 y 5, ya que la estación origen solo estará lista para recibir otro palet cuando la estación destino detecte la llegada del palet que estaba en movimiento.

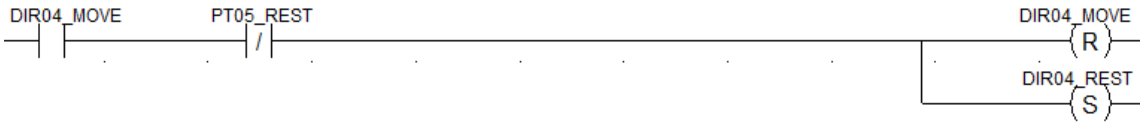


Figura 93 - Línea Ladder desactivación Move sin sensor retenedor

En este caso, la desactivación del estado avance se producirá cuando el palet llegue a la estación destino pero el estado reposo de la estación origen se activará cuando el sensor de esta estación deje de detectar palet y será posible la llegada de otro palet a la estación destino. Permite la combinación de estados 4 y 5.

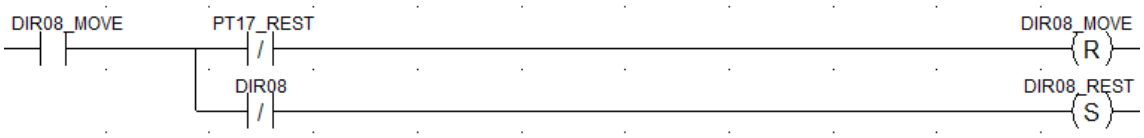


Figura 94 - Línea Ladder desactivación Move con sensor retenedor

En el caso de una intersección con dos entradas y una salida como en la siguiente imagen, se tiene que tener en cuenta que no se pueda producir dos movimientos de entrada en la plataforma.

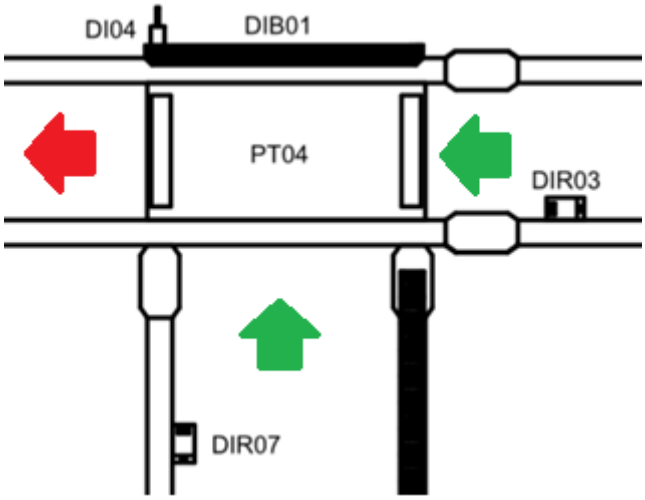


Figura 95 - Flujo plataforma con dos entradas y una salida

Esto se consigue colocando en la línea de código Ladder de cada estación de entrada, el MOVE negado de la otra estación de entrada de la plataforma.

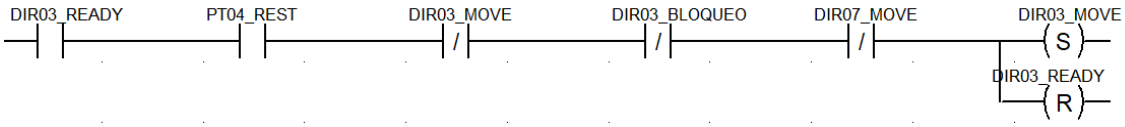


Figura 96 - Línea Ladder activación Move plataforma dos entradas y una salida

En el caso que las dos estaciones estén en estado petición a la espera de la entrada de un palet en la plataforma, se ha programado el sistema para que cada vez que se produzca esta situación la entrada sea de una estación diferente. Se ha realizado en la sección PRIORIDADES.

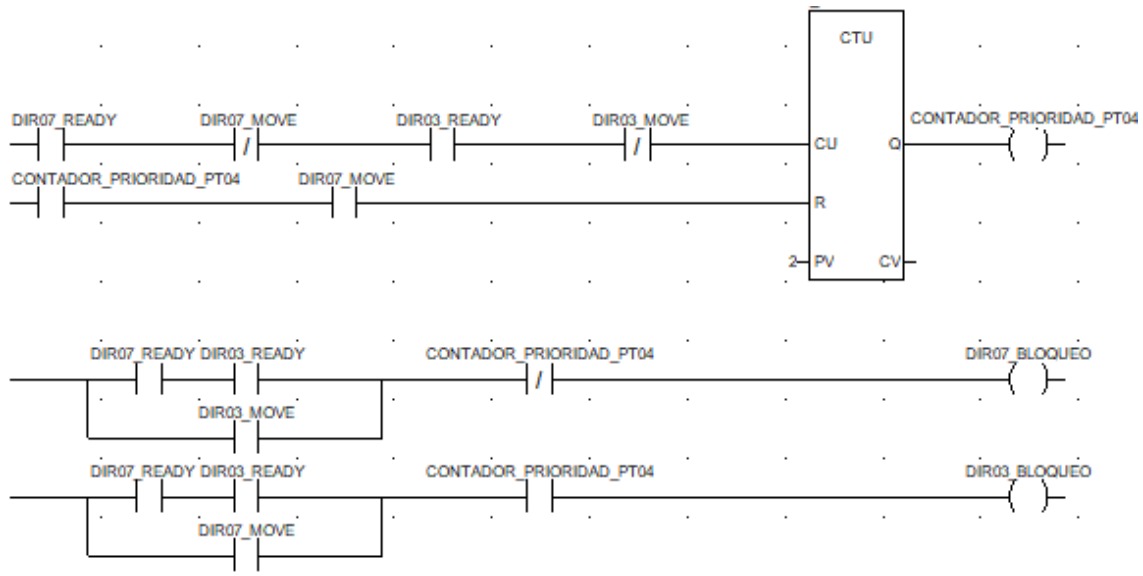


Figura 97 - Líneas Ladder prioridades

La señal de salida de los retenedores está en función del estado activado de avance de la estación y la desactivación del estado reposo y petición. Si el avance está activado y los otros

dos estados desactivados, el pistón del retenedor estará bajado. En cambio, si el estado avance está desactivado o alguno de los estados reposo o petición está activado, el pistón estará levantado. Esto ocurrirá de esta manera si no tenemos en cuenta el temporizador TOF.

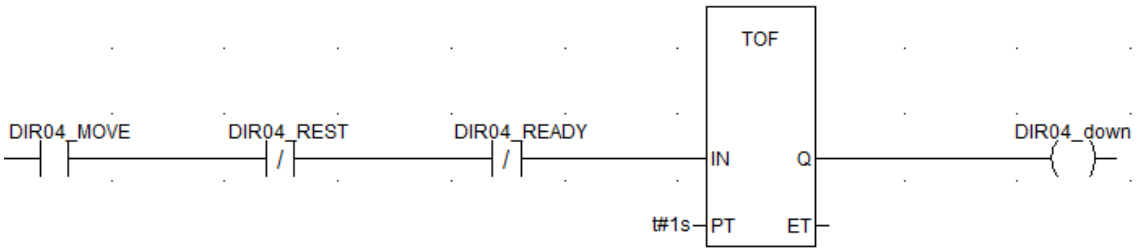


Figura 98 - Señal salida retenedores

La función del TOF es realizar un retraso de la desconexión de la señal de salida consiguiendo que cuando la señal de entrada al temporizador se desactive, la señal de salida esté 1 segundo activada (pistón del retenedor estará un segundo más bajado).

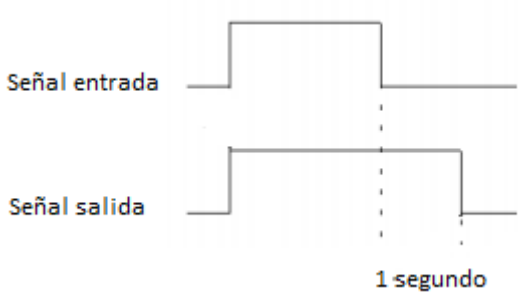


Figura 99 - Cronograma señal entrada y salida TOF

La señal de salida ejemplo del programa es DIR04_down que está direccionada a la salida del sensor DIR04 de la línea. Se realiza el direccionamiento para todos los retenedores y sus señales de salida.

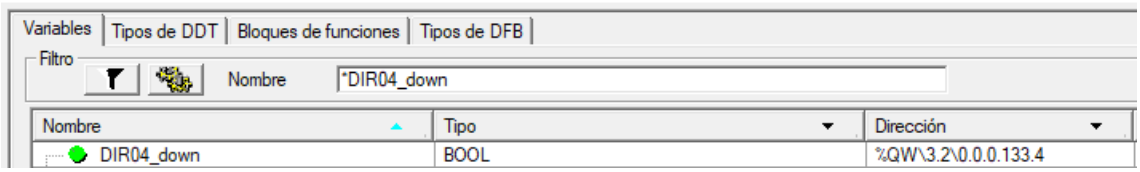


Figura 100 - Direccionamiento señal salida retenedor

La estación DIR06 es una excepción porque el retenedor está en el PLC del pulmón. Se envía la información del sensor a la línea CAN y la línea del pulmón lee la información de la señal de salida de la estación del sensor DIR06 de la línea CAN para subir o bajar el pistón de este sensor.

En las siguientes imágenes se observa el envío del estado del sensor (%MW46) (detecta o no detecta Palet) del pulmón a línea CAN. La línea del pulmón lee el estado de la señal de salida de la estación DIR06 (%MW10) procedente de la línea CAN.

Dirección IP	ID de unidad	Timeout de estado funcional	Velocidad de repetición (ms)	Leer objeto maestro	Leer índice de esclavo	Leer longitud	Último valor (Entrada)	Escribir objeto maestro	Escribir índice de esclavo	Escribir longitud
147.83.74.187	255	1500	60	%MW10	10	1	Mantener último	%MW46	46	1

Figura 101 - Comunicación PT15 REST y PT08 REST

SECCIÓN COMUNICACIÓN	
PULMÓN	CAN
%MW46.0 := DIR06; DIR06_DOWN := %MW10.0;	DIR06 := %MW46.0; %MW10.0 := DIR06_DOWN;

Figura 102 - Sección comunicación PT15 REST y PT08 REST

Las plataformas siguen el modelo de programación de los retenedores pero sin el temporizador TOF. En unos casos la señal activará la subida y en otros la bajada de la plataforma.

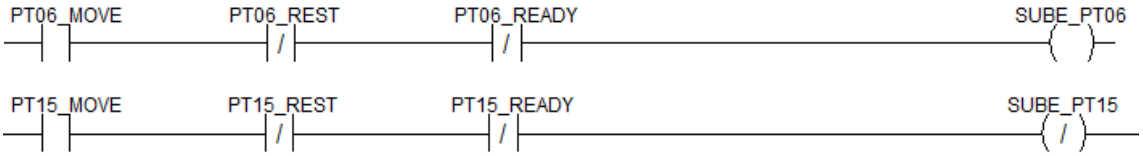


Figura 103 - Señal de salida plataforma

También se direcciona la señal de salida del programa con el elemento físico de la línea.

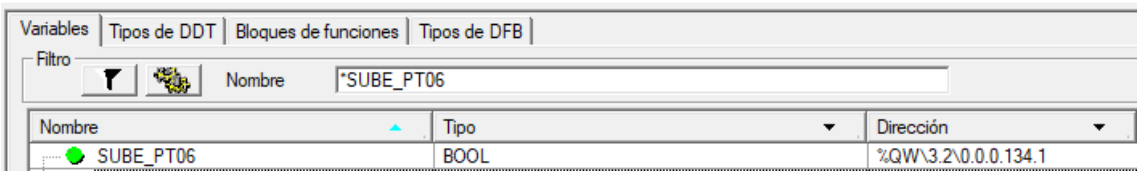


Figura 104 - Direccionamiento señal de salida con plataforma

4.7 Motores

Para la activación de los diferentes motores de la línea y plataformas se utilizan los estados de avance oportunos.

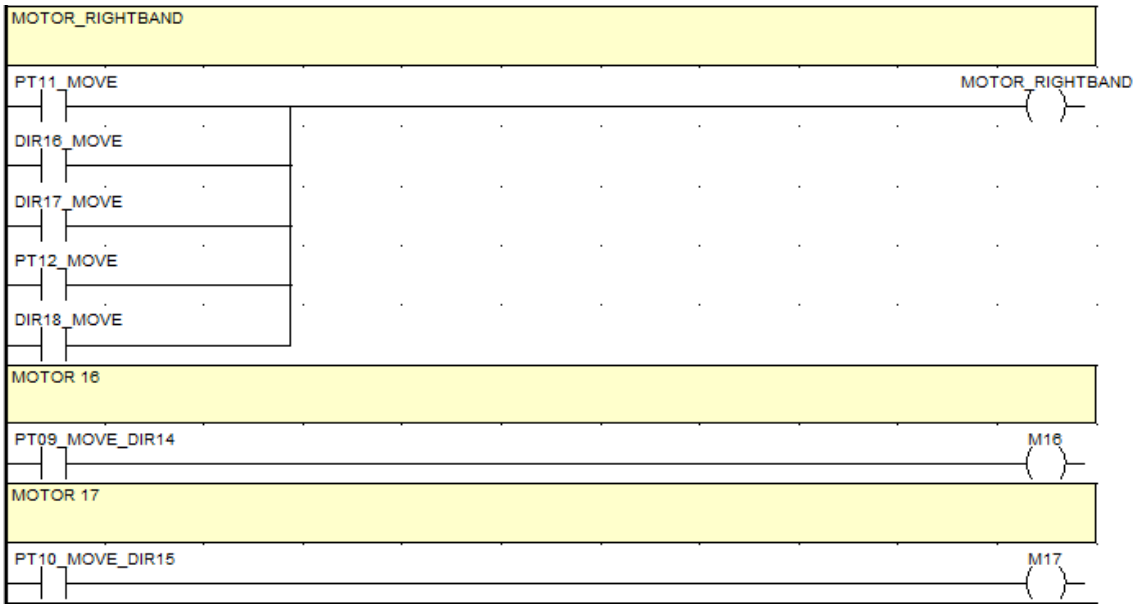


Figura 105 - Ladder motores

Ilustración de la línea con los nombres de los motores actualizados según el programa de este proyecto.

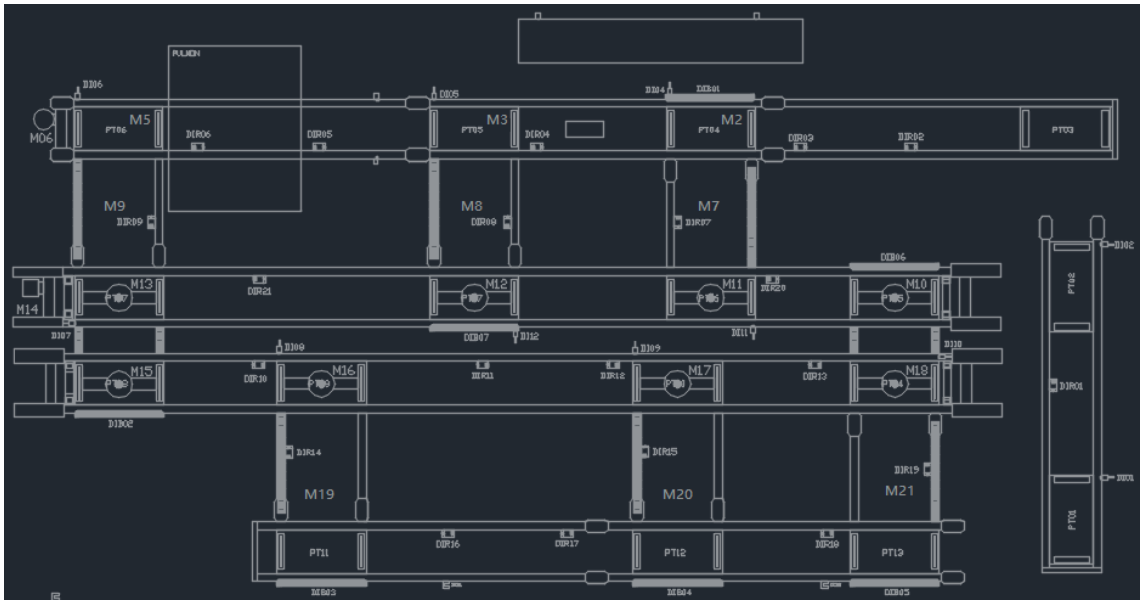


Figura 106 - Motores actualizados código de programación

4.8 Trazabilidad de la celda industrial

Para tener conocimiento de lo que está sucediendo en la línea se debe saber cuál es la trazabilidad que se define como una serie de procedimientos que permiten seguir el proceso de evolución de un producto en cada una de sus etapas.

En esta simulación se puede dividir en dos modelos. Uno es conocer el tipo de producto con su id de producción y el id del palet en el que se está transportando que se consigue guardando estos datos en diferentes tablas del programa. Estas tablas tienen 10 posiciones y en cada una hay una estructura llamada PRODUCTO. Esta estructura está formada por 3 variables tipo INT llamadas ID, TIPO_PRODUCTO y BANDEJA.

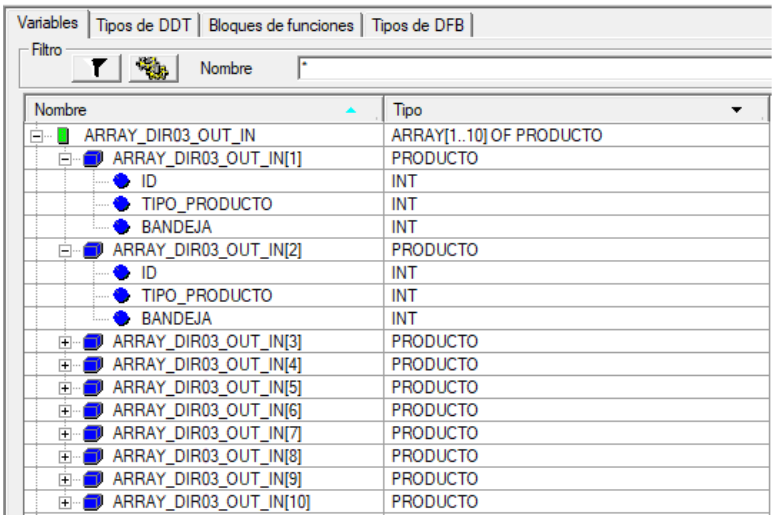


Figura 107 - Tabla trazabilidad tipo producto

Estas tablas se utilizan para traspasar información entre ellas para no perderla cuando hay una intersección en la línea, como se puede ver en la ilustración de la línea Profibus:

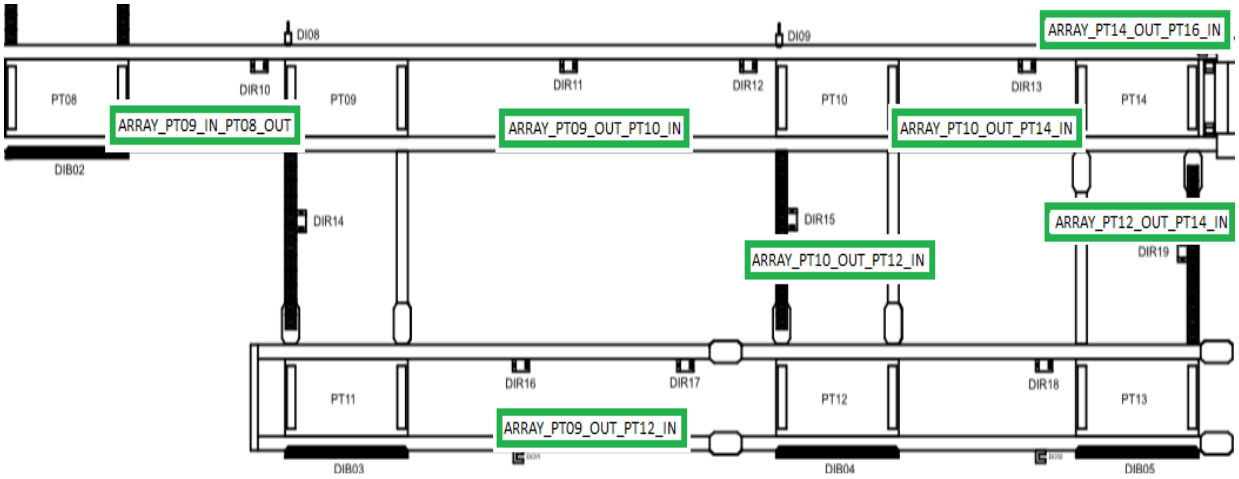


Figura 108 - Distribución tablas trazabilidad tipo producto (Profibus)

Por ejemplo, en la intersección de la plataforma 9, la información de la tabla ARRAY_PT09_IN_PT08_OUT puede ser traspasada a dos tablas. La selección de la tabla depende de la dirección que tenga el palet. Si este va hacía la plataforma 10 la información de la tabla anterior será guardada en ARRAY_PT09_OUT_PT10_IN. En cambio, si va hacía la

plataforma 12, se guardará en la tabla ARRAY_PT09_OUT_PT12_IN. El intercambio de datos se realiza gracias a 3 contadores que hay en cada una de las intersecciones que nos permite leer y escribir correctamente las tablas de las intersecciones. En el ejemplo anterior, los contadores y tablas seguirían esta distribución:

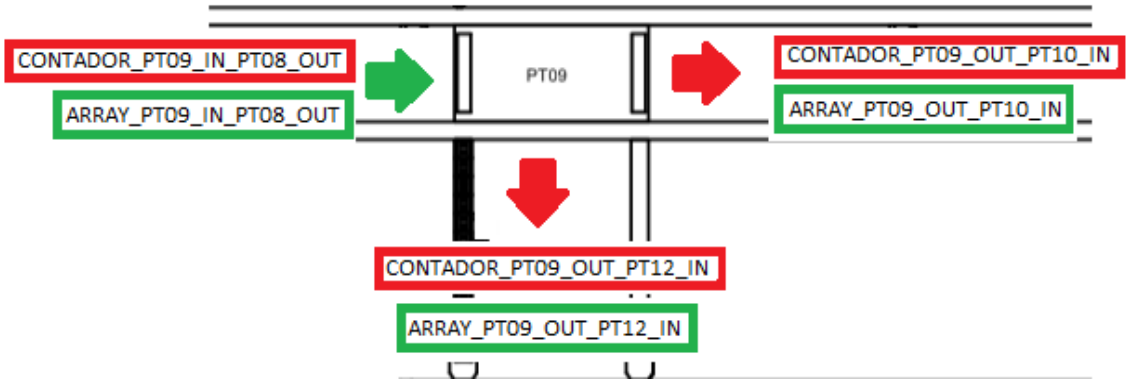


Figura 109 - Contadores y tablas plataforma ejemplo

El código de programación que realiza este traspaso está ubicado en la sección llamada ARRAYS_DIRECCIONES. Cada vez que hay un flanco ascendente en alguno de los retenedores asociados a la plataforma, se produce un aumento del contador asociado a este y un traspaso de información si es necesario.

```
(*PT09 ARRAY*)
DIR10_MOVE_E := DIR10_MOVE;
DIR10_MOVE_F_A := RE (DIR10_MOVE_E);

PT09_MOVE_DIR14_E := PT09_MOVE_DIR14;
PT09_MOVE_DIR11_E := PT09_MOVE_DIR11;

PT09_MOVE_DIR14_F_A := RE (PT09_MOVE_DIR14_E);
PT09_MOVE_DIR11_F_A := RE (PT09_MOVE_DIR11_E);

IF (DIR10_MOVE_F_A) = TRUE THEN

        INC (CONTADOR_PT09_IN_PT08_OUT);

END_IF;

IF (PT09_MOVE_DIR14_F_A) = TRUE THEN

        INC (CONTADOR_PT09_OUT_PT12_IN);

        ARRAY_PT09_OUT_PT12_IN[CONTADOR_PT09_OUT_PT12_IN] := ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT09_IN_PT08_OUT];

END_IF;

IF (PT09_MOVE_DIR11_F_A) = TRUE THEN

        INC (CONTADOR_PT09_OUT_PT10_IN);

        ARRAY_PT09_OUT_PT10_IN[CONTADOR_PT09_OUT_PT10_IN] := ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT09_IN_PT08_OUT];

END_IF;
```

Figura 110 - Traspaso información (Plataforma PT09 ejemplo)

El contador, al llegar al límite de la tabla (10) se le vuelve a asignar el valor 0 para poder recorrer otra vez la tabla.

```
(*RESET CONTADORES*)

IF CONTADOR_PT09_IN_PT08_OUT = 10 THEN

        CONTADOR_PT09_IN_PT08_OUT := 0;

END_IF;
```

Figura 111 - Reset contador

En las uniones entre la línea CAN y la PROFIBUS se necesita hacer un intercambio de datos entre las tablas. Esto se produce de la plataforma 7 a la 8 (ARRAY_PT07_OUT_PT08_IN -> ARRAY_PT09_IN_PT08_OUT) y de la plataforma 14 a la 16 (ARRAY_PT14_OUT_PT16_IN -> ARRAY_PT14_OUT_PT16_IN).

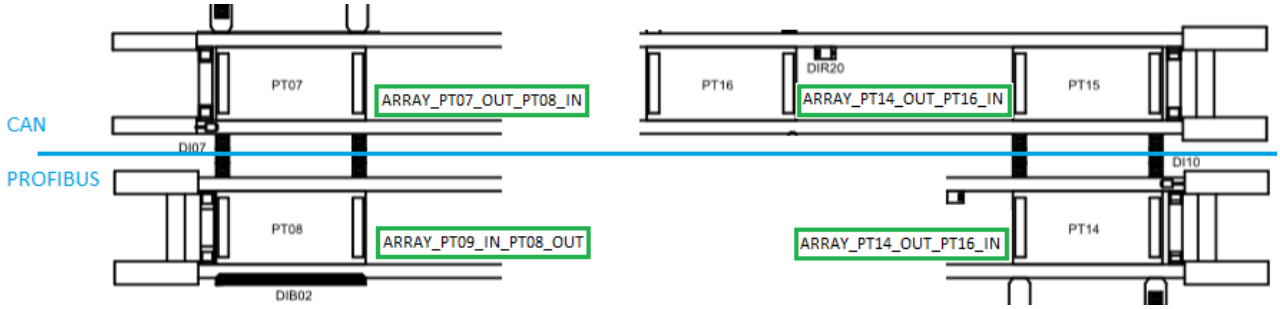


Figura 112 - Uniones CAN - Profibus

El traspaso de información se realiza a partir del módulo de comunicaciones del programa de la línea PROFIBUS en el que se leen los datos de la tabla ARRAY_PT07_OUT_PT08_IN que está en la dirección de memoria %MW305 del programa CAN y se copian en la dirección de memoria %MW305 (ARRAY_PT09_IN_PT08_OUT) del programa PROFIBUS. Los datos de la dirección de memoria %MW352 de la línea PROFIBUS (ARRAY_PT14_OUT_PT16_IN) se escribe en la dirección de memoria %MW402 de la línea CAN (ARRAY_PT14_OUT_PsT16_IN).

Dirección IP	ID de unidad	Timeout de estado funcional	Velocidad de repetición (ms)	Leer objeto maestro	Leer índice de esclavo	Leer longitud	Último valor (Entrada)	Escribir objeto maestro	Escribir índice de esclavo	Escribir longitud
147.83.74.187	255	1500	60	%MW305	305	30	Establecer en 0	%MW352	402	30

Figura 113 - Comunicación Arrays unión CAN - Profibus

Línea PROFIBUS:

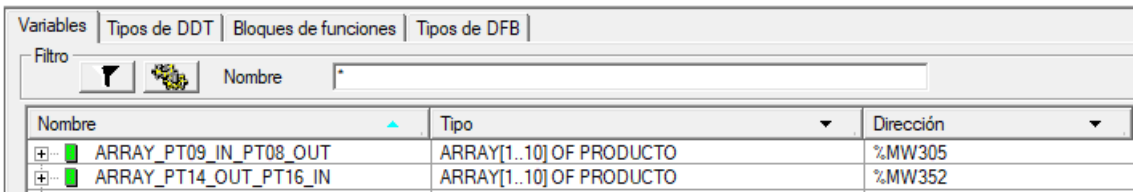


Figura 114 - Direccionamiento arrays Profibus

Línea CAN:

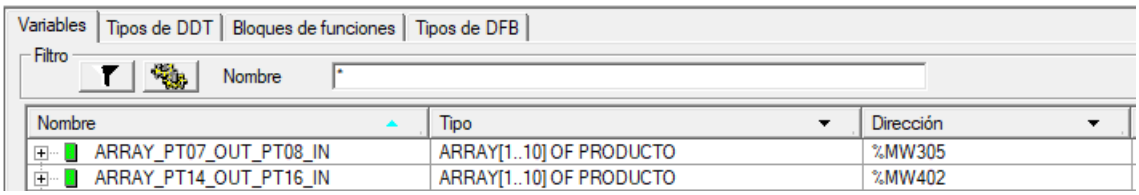


Figura 115 - Direccionamiento arrays CAN

El otro modelo está relacionado con los datos de las diferentes estaciones de trabajo de la línea que se guardan en otras tablas del programa con una longitud de 4 filas. En estas tablas hay una estructura formada por una variable tipo tiempo para almacenar el tiempo transcurrido en la estación de trabajo y una variable tipo INT para relacionar el tiempo con el ID del producto correspondiente.

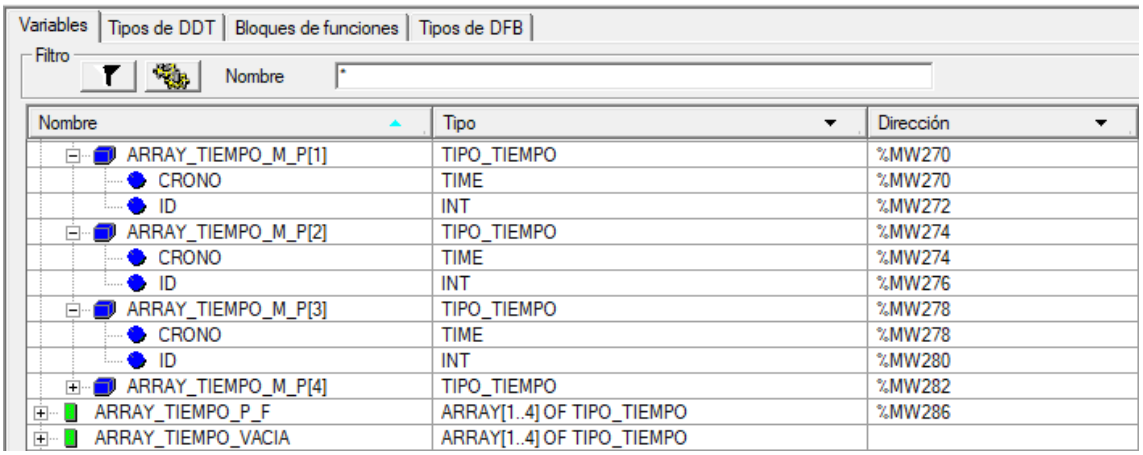


Figura 116 - Tabla trazabilidad tipo tiempo

4.9 Dashboard Node-red

El dashboard de Node-red permite darle valor y enviar variables a los PLCs. Estas variables que se selecciona el valor o activan son los siguientes:

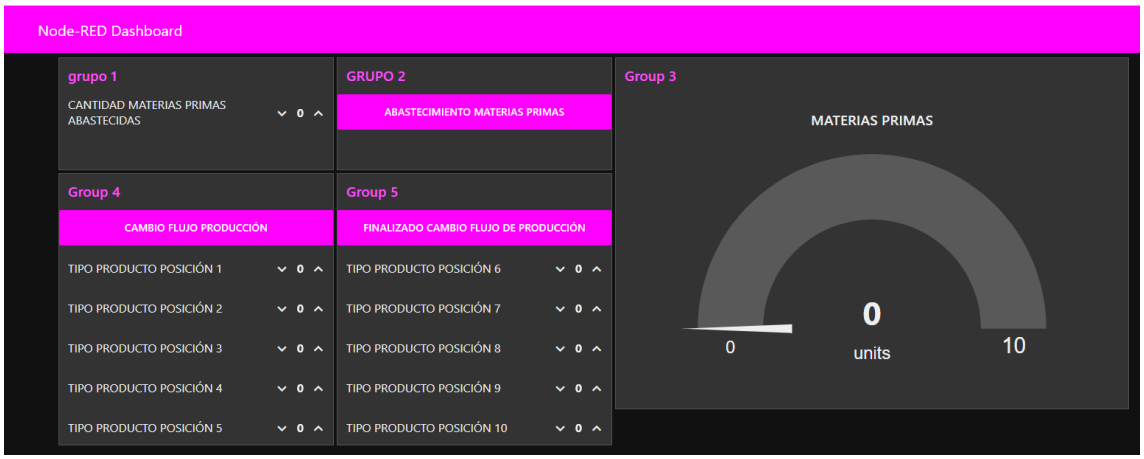


Figura 117 - Dashboard Node red

A cada nodo de función se le asigna la dirección correspondiente a la variable de la línea CAN, asociada al valor seleccionado en el dashboard.

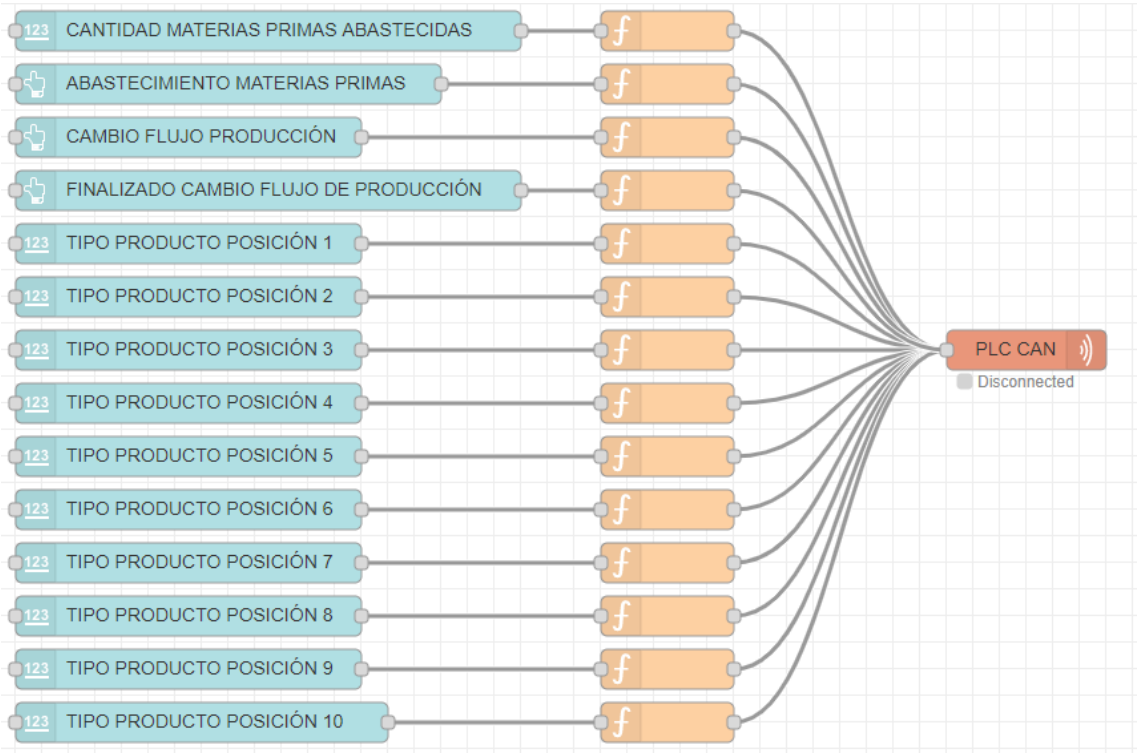


Figura 118 -Nodos dashboard envió datos PLC CAN

Cantidad Materias Primas Abastecidas: Selecciona la cantidad a través de un nodo numérico que nos permite aumentar o disminuir el valor. Esta variable es de tipo INT y está enlazada con la dirección de memoria %MW104 del PLC de la línea CAN.

El siguiente diagrama de flujo Node-red recibe información del PLC de la línea CAN.

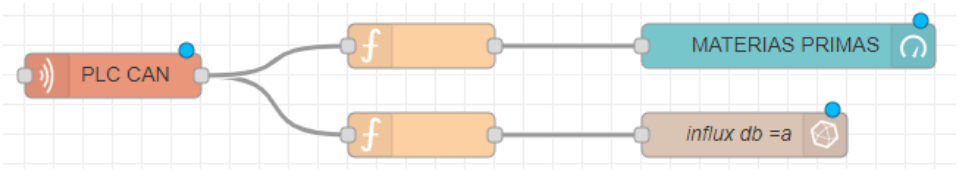


Figura 119 – Nodos reciben datos PLC CAN

Esta bifurcación del diagrama de flujo nos permite leer el valor de la variable %MW263 del PLC (CAN). Esta variable es CONTADOR_CANTIDAD_M_PRIMAS que nos suministra el total de materias primas existente en la célula industrial.

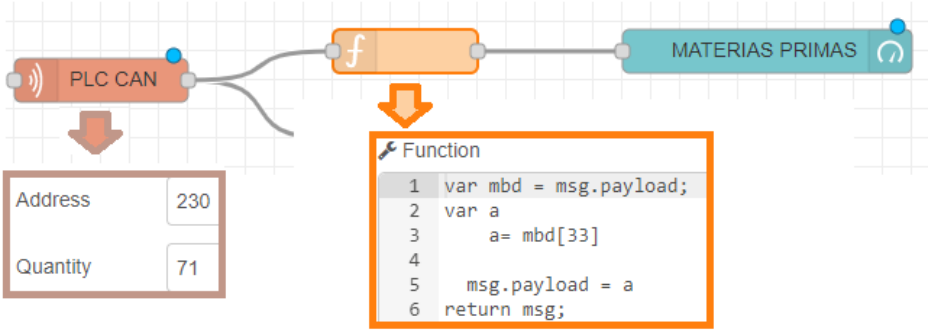


Figura 120 - Nodo dashboard recibe datos PLC CAN

El valor de la variable es enviado a un nodo de Node-red que nos muestra un gráfico en el dashboard con este valor.

Figura 121 – Variable CONTADOR_CANTIDAD_M_PRIMAS

Figura 122 - Diferentes estados widget tipo velocímetro

CANTIDAD_M_PRIMAS: Selecciona la cantidad de materias primas abastecidas a través de un nodo numérico que nos permite aumentar o disminuir el valor. Está direccionado a la variable CANTIDAD_M_PRIMAS_ABASTECIDAS en la posición %MW104 (PLC CAN).

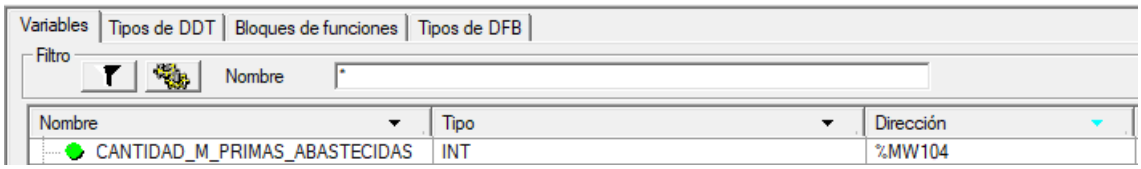


Figura 123 – Variable CANTIDAD_M_PRIMAS_ABASTECIDAS

Abastecimiento Materias Primas: Al pulsar este Botón activamos la variable booleana ABASTECIMIENTO_M_PRIMAS de la programación de la línea CAN.

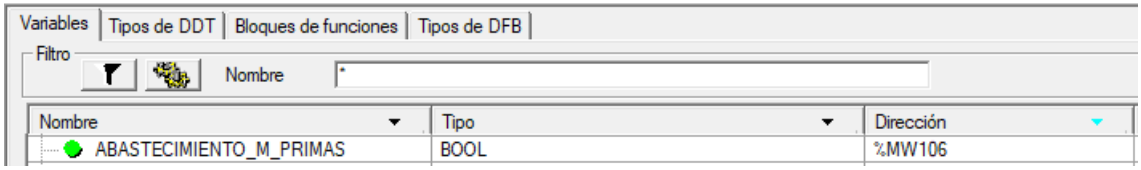


Figura 124 – Variable ABASTECIMIENTO_M_PRIMAS

Esta programación permite simular el suministro de materias primas a la zona de suministro de la línea, como podría ser un palet de madera con su cargamento. La variable ABASTECIMIENTO_M_PRIMAS procedente de Node-red nos informa de la realización del suministro de materias primas. Podría ser una señal procedente de un PLC, pulsador, botón, etc. Si no se realiza la activación de esta variable no se producirá el sumatorio de las materias primas.

```
(*ABASTECIMIENTO MATERIAS PRIMAS*)

IF ABASTECIMIENTO_M_PRIMAS = TRUE THEN

CONTADOR_CANTIDAD_M_PRIMAS := CONTADOR_CANTIDAD_M_PRIMAS + CANTIDAD_M_PRIMAS_ABASTECIDAS;

ABASTECIMIENTO_M_PRIMAS := FALSE;

END_IF;
```

Figura 125 – Código ST Abastecimiento Materias Primas

Tipo producto posición _: Selecciona el tipo de producto a través de un nodo numérico que nos permite aumentar o disminuir el valor. Cada nodo numérico está direccionado a una posición de la tabla de trazabilidad ARRAY_PT06_CARGA_M_PRIMAS (PLC CAN) que nos permite cambiar el valor del tipo producto en 10 casos.

Ejemplo nodo Tipo producto posición 1:

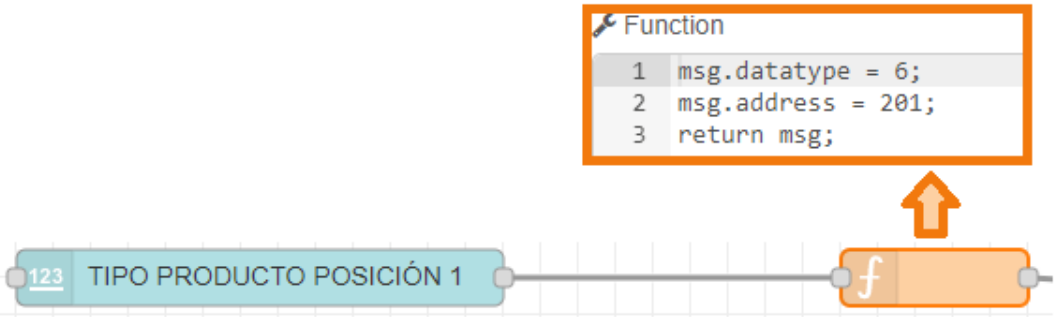


Figura 126 - Dirección PLC (CAN) enviada por Node-red (201)

Variables | Tipos de DDT | Bloques de funciones | Tipos de DFB

Filtro — Nombre: *

Nombre	Tipo	Dirección
ARRAY_PT06_CARGA_M_PRIMAS	ARRAY[1..10] OF PRODUCTO	%MW200
ARRAY_PT06_CARGA_M_PRIMAS[1]	PRODUCTO	%MW200
ID	INT	%MW200
TIPO_PRODUCTO	INT	%MW201
BANDEJA	INT	%MW202
ARRAY_PT06_CARGA_M_PRIMAS[2]	PRODUCTO	%MW203
ARRAY_PT06_CARGA_M_PRIMAS[3]	PRODUCTO	%MW206
ARRAY_PT06_CARGA_M_PRIMAS[4]	PRODUCTO	%MW209
ARRAY_PT06_CARGA_M_PRIMAS[5]	PRODUCTO	%MW212
ARRAY_PT06_CARGA_M_PRIMAS[6]	PRODUCTO	%MW215
ARRAY_PT06_CARGA_M_PRIMAS[7]	PRODUCTO	%MW218
ARRAY_PT06_CARGA_M_PRIMAS[8]	PRODUCTO	%MW221
ARRAY_PT06_CARGA_M_PRIMAS[9]	PRODUCTO	%MW224
ARRAY_PT06_CARGA_M_PRIMAS[10]	PRODUCTO	%MW227

Figura 127 – Dirección TIPO_PRODUCTO de la primera posición en la tabla PT06_CARGA_M_PRIMAS *(%MW201)*

Al variar directamente estos nodos en el dashboard varían los valores de la memoria del PLC.

Se ha tenido en cuenta la posibilidad de realizar un paro de la estación PT06 al realizar un cambio de flujo de producción (cambio tipo producto).

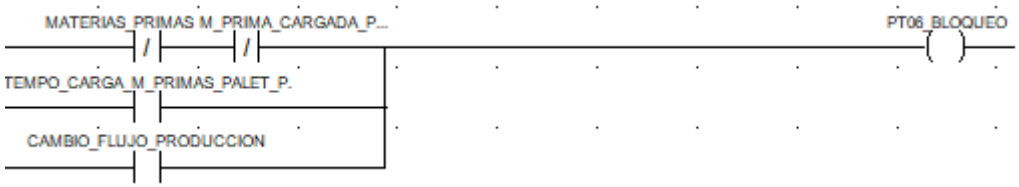


Figura 128 - Ladder cambio flujo de producción

Se ha realizado utilizando los botones del dashboard de Node-red CAMBIO FLUJO PRODUCCIÓN y FINALIZADO CAMBIO FLUJO DE PRODUCCIÓN.

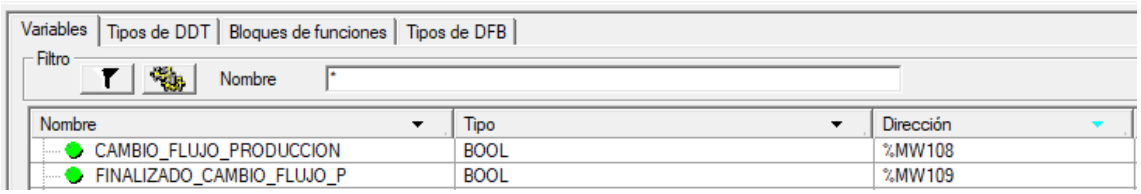


Figura 129 – Variables Cambio flujo producción y finalizado cambio flujo de producción

Cuando se clica CAMBIO FLUJO PRODUCCIÓN (tipo de producto guardado en la tabla) se bloquea la salida de Palets en la estación PT06 hasta que finalice el cambio de flujo (clicar al botón FINALIZADO CAMBIO FLUJO DE PRODUCCIÓN).

```
(*CAMBIO FLUJO DE PRODUCCIÓN*)

IF CAMBIO_FLUJO_PRODUCCION = TRUE AND FINALIZADO_CAMBIO_FLUJO_P = TRUE THEN

        CAMBIO_FLUJO_PRODUCCION := FALSE;
        FINALIZADO_CAMBIO_FLUJO_P := FALSE;

END_IF;
```

Figura 130 - Código ST cambio flujo de producción

4.10 Estaciones de trabajo

En la simulación del proceso industrial realizada en este proyecto se han diseñado 5 estaciones de trabajo en las que podría haber robots o operarios.

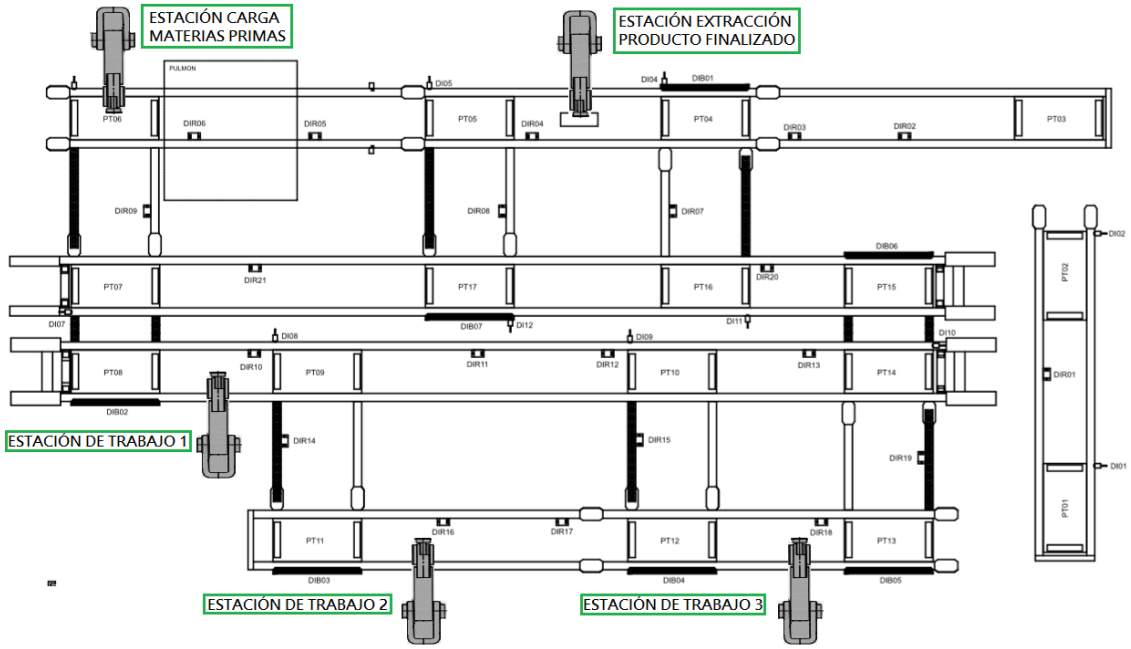


Figura 131 - Distribución estaciones de trabajo celda industrial

La primera es la estación de carga de las materias primas, en la que se realiza la carga de materias primas en los palets.

La estación realizará esta actividad siempre que haya un palet vacío en la estación de la línea PT06 y tenga materias primas disponibles. El temporizador TP simula el tiempo de realización de la actividad en la Sección ESTACIONES_DE_TRABAJO.

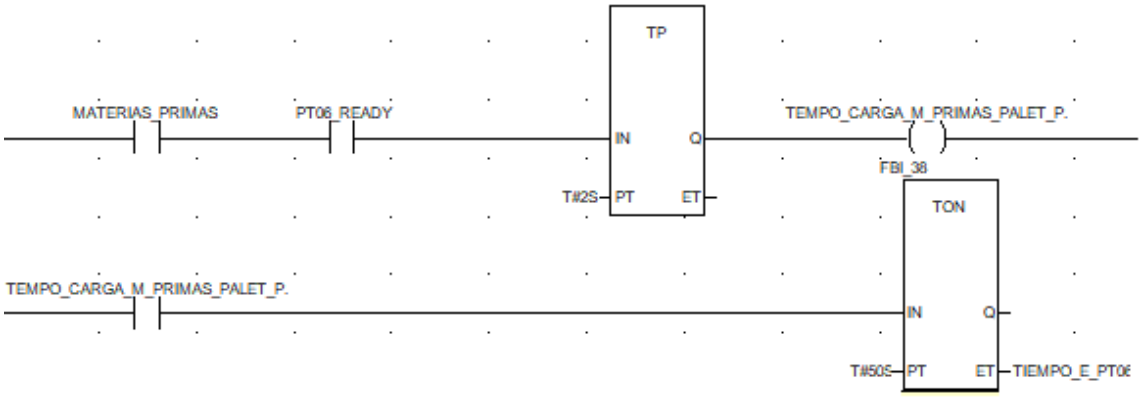


Figura 132 - Ladder estación carga materias primas

Temporizador TON calcula el tiempo de realización de la tarea (TIEMPO_E_PT06) y se guarda en una variable llamada TIEMPO_E_M_P en la sección CRONOMETROS_ESTACIONES. El valor final es el tiempo máximo que ha estado activada la estación (TEMPO_CARGA_M_PRIMAS_PALET_PT06).

```
IF TEMPO_CARGA_M_PRIMAS_PALET_PT06 = TRUE THEN

        TIEMPO_E_M_P := TIEMPO_E_PT06;
        FLANCO_D_E_M_P := FALSE;
END_IF;
```

Figura 133 - Sección CRONOMETRO_ESTACIONES

El bloqueo del movimiento de la estación PT06 se produce cuando no hay materias Primas, se están cargando estas o se ha producido un cambio de flujo de producción.

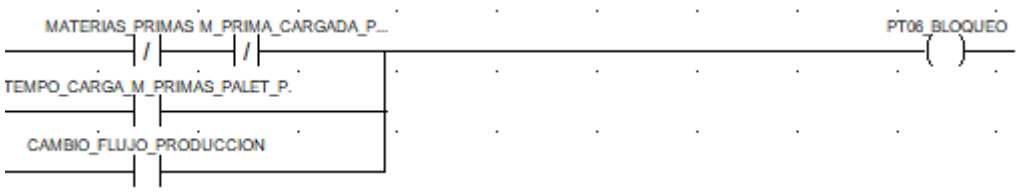


Figura 134 – Ladder Bloqueo PT06

En la sección ESTACIONES_DE_TRABAJO_ST, la siguiente programación nos permite saber si hay suficientes materias primas para suministrar a la línea.

```
(*INDICADOR DE LA EXISTENCIA DE MATERIAS PRIMAS EN LA LINEA*)

IF (CONTADOR_CANTIDAD_M_PRIMAS > 0) THEN

        MATERIAS_PRIMAS := TRUE;

ELSE

        MATERIAS_PRIMAS := FALSE;

END_IF;
```

Figura 135 – Código ST Indicador existencia materias primas en la línea

Esta otra programación permite simular el suministro de materias primas a la zona de suministro de la línea, como podría ser un palet de madera con su cargamento. La variable ABASTECIMIENTO_M_PRIMAS nos informa de la realización del suministro de materias primas. Podría ser una señal procedente de un PLC, pulsador, botón, etc. En nuestro caso procede de Node-red.

```
(*ABASTECIMIENTO MATERIAS PRIMAS*)

IF ABASTECIMIENTO_M_PRIMAS = TRUE THEN

CONTADOR_CANTIDAD_M_PRIMAS := CONTADOR_CANTIDAD_M_PRIMAS + CANTIDAD_M_PRIMAS_ABASTECIDAS;

ABASTECIMIENTO_M_PRIMAS := FALSE;

END_IF;
```

Figura 136 - Código ST Abastecimiento materias primas

El siguiente código de programación, una vez empieza la carga de materias primas en la estación PT06, indica que se está realizando esta acción y que se reduce la cantidad de materias primas existente en la línea.

```
(*DISMINUIR CANTIDAD MATERIAS PRIMAS DE LA LINEA*)

IF (RE (TEMPO_CARGA_M_PRIMAS_PALET_PT06)) = TRUE THEN
        M_PRIMA_CARGADA_PT06 := TRUE;
        IF (CONTADOR_CANTIDAD_M_PRIMAS > 0)THEN
        DEC (CONTADOR_CANTIDAD_M_PRIMAS);
        END_IF;

END_IF;
```

Figura 137 - Código ST disminuir cantidad materias primas de la línea

Cuando termina la carga de materias primas en la estación PT06, esta función nos indica la finalización a partir de la variable M_PRIMA_CARGADA_PT06. A continuación procede a incrementar el contador que utilizaremos para guardar información en la tabla de trazabilidad creada para esta estación (ARRAY_PT06_CARGA_M_PRIMAS) y que en otras líneas de código se utilizarán para traspasar la información a la tabla de trazabilidad de la línea ARRAY_PT07_OUT_PT08_IN.

La información almacenada en ARRAY_PT06_CARGA_M_PRIMAS se realiza en la sección ARRAYS_DIRECCIONES. Es el valor del id del producto que incrementa una unidad cada vez que se ha cargado una materia prima en la estación PT06 y el ID del palet (variable BANDEJA) procedente de la tabla anterior ARRAY_PT05_OUT_PT06_IN.

```
IF (FE(TEMPO_CARGA_M_PRIMAS_PALET_PT06)) = TRUE THEN
        M_PRIMA_CARGADA_PT06 := FALSE;
        INC (CONTADOR_CARGA_M_PRIMAS);
        INC (CONTADOR_ID);
        INC (CONTADOR_TIEMPO_M_P);
        ARRAY_PT06_CARGA_M_PRIMAS[CONTADOR_CARGA_M_PRIMAS].ID:= CONTADOR_ID;
        ARRAY_PT06_CARGA_M_PRIMAS[CONTADOR_CARGA_M_PRIMAS].BANDEJA := ARRAY_PT05_OUT_PT06_IN[CONTADOR_CARGA_M_PRIMAS].BANDEJA;

        IF CONTADOR_TIEMPO_M_P = 5 THEN
        ARRAY_TIEMPO_M_P := ARRAY_TIEMPO_VACIA;
        CONTADOR_TIEMPO_M_P := 1;
        END_IF;

        ARRAY_TIEMPO_M_P[CONTADOR_TIEMPO_M_P].CRONO := TIEMPO_E_M_P;
        ARRAY_TIEMPO_M_P[CONTADOR_TIEMPO_M_P].ID := CONTADOR_ID;

        FLANCO_D_E_M_P := TRUE;
END_IF;
```

Figura 138 - Traspase información tablas estación PT06

La función IF también realiza el almacenaje del tiempo transcurrido en la carga de materias primas para cada id de producto. Cuando el contador de recorrida de esta tabla es igual a 5, se le da el valor “1” para no salirse de la longitud de esta tabla y se le da el valor “true” a la variable FLANCO_D_E_M_P utilizada para la programación en node red.

Los palets cargados con las materias primas van después a las estaciones de trabajo 1, 2 o 3, en las que dependiendo del tipo de producto leído en la tabla de trazabilidad realizarán un recorrido en la línea pasando por las estaciones de trabajo que corresponden a cada producto. En la estación PT09 del proyecto se observa que cuando esta estación está en modo petición y dependiendo del tipo producto, se selecciona una dirección u otra. Esto sucede en todas las plataformas con una entrada y dos salidas.

```
(*DIRECCIONES INTERCEPCIONES*)
(*PT09*)
IF PT09_READY = TRUE AND ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT09_IN_PT08_OUT].TIPO_PRODUCTO = 0 THEN

PT09_DIRECCION_DIR11 := TRUE;

END_IF;

IF PT09_READY = TRUE AND ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT09_IN_PT08_OUT].TIPO_PRODUCTO = 1 THEN

PT09_DIRECCION_DIR14 := TRUE;

END_IF;

IF PT09_READY = TRUE AND ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT09_IN_PT08_OUT].TIPO_PRODUCTO = 2 THEN

PT09_DIRECCION_DIR11 := TRUE;

END_IF;

IF PT09_READY = TRUE AND ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT09_IN_PT08_OUT].TIPO_PRODUCTO = 3 THEN

PT09_DIRECCION_DIR11 := TRUE;

END_IF;
```

Figura 139 - Código ST direcciones intercepciones

Las estaciones de trabajo 1, 2 y 3 siguen el mismo tipo de código de programación que se explica en la estación de trabajo 1. El primer paso es reconocer el tipo de producto que lleva el palet antes de entrar en la estación de trabajo, para saber si se tiene que realizar la actividad de esa estación. Si es así, se activa la variable TRABAJO_ESTACION_1.

```
(*ESTACION DE TRABAJO 1*)

PT08_MOVE_E := PT08_MOVE;

PT08_MOVE_F_A := RE (PT08_MOVE_E);

IF PT08_MOVE_F_A = TRUE THEN

        INC (CONTADOR_PT08_OUT);

END_IF;

IF PT08_MOVE_F_A = TRUE AND ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT08_OUT].TIPO_PRODUCTO > 0 THEN

        TRABAJO_ESTACION_1 := TRUE;

END_IF;
```

Figura 140 - Código ST estación 1

Una vez esta variable está activada, la estación no empezará a realizar su actividad hasta que el retenedor de su estación esté en estado de petición. El temporizador TP simula el tiempo de realización de la actividad y el TON nos permite contar el tiempo durante el que está activa la estación TIEMPO_DIR10.

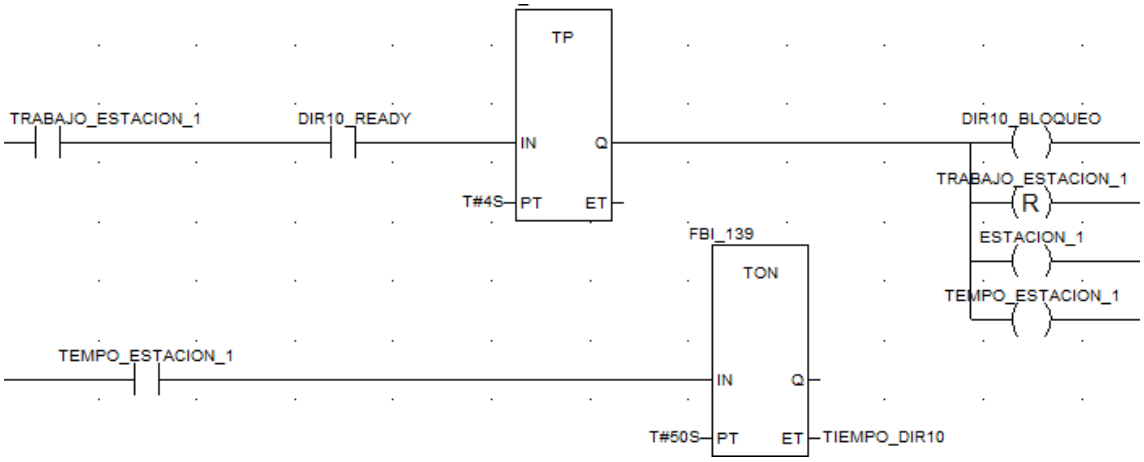


Figura 141 - Ladder estación 1

Este tiempo se va guardado y almacenando en la variable TIEMPO_E_1 siempre que la estación esté activa. Esto permite almacenar el tiempo máximo de actividad en la variable anterior. La variable FLANCO_D_E_1 se utiliza en la programación del software Node red.

```
IF TEMPO_ESTACION_1 = TRUE THEN

        TIEMPO_E_1 := TIEMPO_DIR10;
        FLANCO_D_E_1 := FALSE;
END_IF;
```

Figura 142 - Cronometro estaciones

Se incrementa el CONTADOR_E_1 que indica la cantidad de productos realizados en la estación 1 con la información de la línea Profibus que es leída por la línea CAN (Flanco activación estación).

```
IF (RE (ESTACION_1_FLANCO)) = TRUE THEN

        INC (CONTADOR_E_1);

END_IF;
```

Figura 143 - Contador estación 1

```
ESTACION_1_FLANCO := %MW351.0;

ESTACION_2_FLANCO := %MW351.1;

ESTACION_3_FLANCO := %MW351.2;
```

Figura 144 – Direccionamiento flancos CAN->Profibus

	Dirección IP	ID de unidad	Timeout de estado funcional	Velocidad de repetición (ms)	Escribir objeto maestro	Escribir índice de esclavo	Escribir longitud
1	147.83.74.187	255	1500	60	%MW351	351	1

Figura 145 – Comunicación Flancos CAN->Profibus

Una vez acabada la actividad en la estación 1 se realiza el traspase del tiempo transcurrido a la tabla ARRAY_TIEMPO_E_1 y el ID del producto del trabajo realizado en la estación.

```
(*CRONOMETRO ESTACION 1*)

TEMPO_ESTACION_1_E := TEMPO_ESTACION_1;

TEMPO_ESTACION_1_F_A := FE(TEMPO_ESTACION_1_E);

IF (TEMPO_ESTACION_1_F_A) = TRUE THEN

        INC(CONTADOR_TIEMPO_E_1);
        IF CONTADOR_TIEMPO_E_1 = 5 THEN
        ARRAY_TIEMPO_E_1 := ARRAY_TIEMPO_VACIA;
        CONTADOR_TIEMPO_E_1 := 1;
        END_IF;

        ARRAY_TIEMPO_E_1[CONTADOR_TIEMPO_E_1].CRONO := TIEMPO_E_1;
        ARRAY_TIEMPO_E_1[CONTADOR_TIEMPO_E_1].ID := ARRAY_PT09_IN_PT08_OUT[CONTADOR_PT08_OUT].ID;

        FLANCO_D_E_1 := TRUE;

END_IF;
```

Figura 146 - Cronometro estación 1

Después de haber pasado por todas las estaciones, el palet llegará a la última estación en la que se extraerá el producto acabado. Esta estación cuenta con un cronómetro que controla el tiempo de realización de la actividad de extracción.

El primer paso es saber si es necesario extraer el producto del palet que va entrar en esta estación. Esto se realiza en la siguiente programación:

```
(*EXTRACCIÓN PRODUCTO FINALIZADO -- RELLENAR TABLA*)


DIR04_READY_E := DIR04_READY;
DIR04_READY_F_A := RE (DIR04_READY_E);

IF PT04_MOVE_F_A = TRUE AND ARRAY_PT04_OUT_PT05_IN[CONTADOR_PT04_OUT].TIPO_PRODUCTO > 0 THEN

        BANDEJA_CON_PRODUCTO := TRUE;

END_IF;
```

Figura 147 - Código ST comprobación producto palet PT04

Si el palet lleva algún producto acabado se procede al traspaso de la trazabilidad del palet a dos tablas. ARRAY_PRODUCTO_FINALIZADO es una tabla de uso interno del programa y sus datos no son borrados ya que se van sobrescribiendo las filas. En cambio, la información de la tabla ARRAY_PRODUCTO_FINALIZADO_ENVIO se envía a la base de datos y al superar el CONTADOR_PRODUCTO_FINALIZADO la longitud de la tabla, se inicializa a "0" todos sus valores para el correcto funcionamiento de la programación de Node red. También se le da el valor "0" a los campos ID y el tipo producto de la tabla ARRAY_PT04_OUT_PT05_IN para vaciar el palet de forma virtual.

```
IF DIR04_READY_F_A = TRUE AND BANDEJA_CON_PRODUCTO = TRUE THEN

        INC(CONTADOR_P_FINALIZADO_ENVIO);
        INC(CONTADOR_PRODUCTO_FINALIZADO);

        IF (CONTADOR_PRODUCTO_FINALIZADO = 11) THEN

        ARRAY_PRODUCTO_FINALIZADO_ENVIO := ARRAY_VACIA;

        CONTADOR_PRODUCTO_FINALIZADO := 1;

        END_IF;

        ARRAY_PRODUCTO_FINALIZADO[CONTADOR_PRODUCTO_FINALIZADO] := ARRAY_PT04_OUT_PT05_IN[CONTADOR_PT04_OUT];
        ARRAY_PRODUCTO_FINALIZADO_ENVIO[CONTADOR_PRODUCTO_FINALIZADO] := ARRAY_PRODUCTO_FINALIZADO[CONTADOR_PRODUCTO_FINALIZAD
        ARRAY_PT04_OUT_PT05_IN[CONTADOR_PT04_OUT].ID := 0;
        ARRAY_PT04_OUT_PT05_IN[CONTADOR_PT04_OUT].TIPO_PRODUCTO := 0;
        BANDEJA_CON_PRODUCTO := FALSE;
```

Figura 148 - - Código ST extracción producto finalizado

Se realiza el reset del CONTADOR_PT04_OUT para el correcto escaneo de la tabla correspondiente.

```
IF CONTADOR_PT04_OUT = 10 THEN

CONTADOR_PT04_OUT := 0;

END_IF;
```

Figura 149 - Reset CONTADOR_PT04_OUT

Se realiza un registro de la cantidad de cada tipo de producto que se extrae de la línea.

```
IF (ARRAY_PRODUCTO_FINALIZADO[CONTADOR_PRODUCTO_FINALIZADO].TIPO_PRODUCTO) = 1 THEN

INC (CONTADOR_T_P_1);

END_IF;

IF (ARRAY_PRODUCTO_FINALIZADO[CONTADOR_PRODUCTO_FINALIZADO].TIPO_PRODUCTO) = 2 THEN

INC (CONTADOR_T_P_2);

END_IF;

IF (ARRAY_PRODUCTO_FINALIZADO[CONTADOR_PRODUCTO_FINALIZADO].TIPO_PRODUCTO) = 3 THEN

INC (CONTADOR_T_P_3);

END_IF;
```

Figura 150 - Contadores cantidad tipo producto

A la hora de escoger la dirección en la plataforma 16 se tiene en cuenta si el palet llega con producto finalizado o si hay algún lugar libre en las estaciones DIR05, DIR06 y PT06 para hacer cola para cargar materias primas. Lo mismo sucede en la estación PT05.

```
IF PT16_READY = TRUE AND ARRAY_PT14_OUT_PT16_IN[CONTADOR_PT16_IN_PT14_OUT].TIPO_PRODUCTO = 0 THEN

        IF (PT06_REST = TRUE OR DIR06_REST = TRUE OR DIR05_REST = TRUE) THEN

        PT16_DIRECCION_DIR07 := TRUE;

        ELSE
        PT16_DIRECCION_PT17 := TRUE;
        END_IF;

END_IF;

IF PT16_READY = TRUE AND ARRAY_PT14_OUT_PT16_IN[CONTADOR_PT16_IN_PT14_OUT].TIPO_PRODUCTO = 1 THEN

PT16_DIRECCION_DIR07 := TRUE;

END_IF;

IF PT16_READY = TRUE AND ARRAY_PT14_OUT_PT16_IN[CONTADOR_PT16_IN_PT14_OUT].TIPO_PRODUCTO = 2 THEN

PT16_DIRECCION_DIR07 := TRUE;

END_IF;

IF PT16_READY = TRUE AND ARRAY_PT14_OUT_PT16_IN[CONTADOR_PT16_IN_PT14_OUT].TIPO_PRODUCTO = 3 THEN

PT16_DIRECCION_DIR07 := TRUE;

END_IF;
```

Figura 151 - Direcciones estación PT16

4.11 Base de datos y paneles de visualización

El primer paso es recibir información de los PLC. El siguiente diagrama de flujo Node-red recibe información del PLC de la línea CAN.

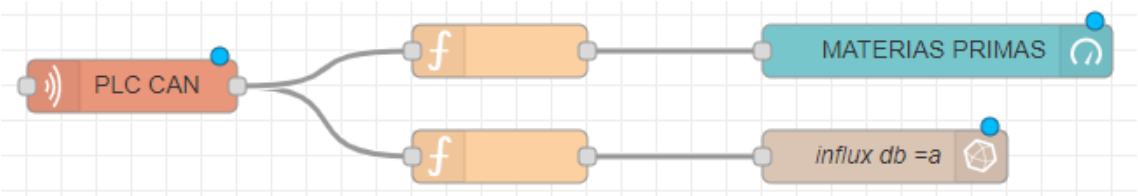


Figura 152 - Diagrama flujo Node red recibe información PLC CAN

Esta bifurcación nos facilita datos del PLC de la línea CAN y después de compilar la función de Node-red, envía los valores a la base de datos llamada 'a' de Influx DB en diferentes measurements.

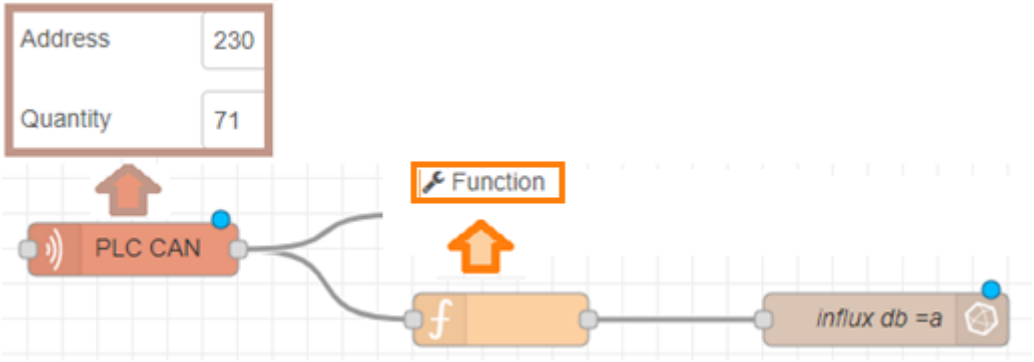


Figura 153 - Bifurcación Node red (PLC->Influx DB)

ModbusTCP entrega los datos en forma de matriz. En cambio, el nodo influxdb está basado en la estructura de datos formada por measurement (se almacenan datos) que podemos comparar a una hoja de Excel con los nombres de las columnas de cada dato. Esto provoca que la unión de estos dos bloques no sea compatible y es necesario insertar una función para adaptar los datos. En este proyecto también se ha utilizado este mismo bloque programándolo con la finalidad de leer los datos de tipo bool en una única posición de memoria del PLC y escribirlos en la base de datos para su uso posterior en el software Grafana. También se ha utilizado para seleccionar datos de las tablas y enviar al nodo de Influx DB los datos requeridos.

Variables | Tipos de DDT | Bloques de funciones | Tipos de DFB

Filtro

Nombre *

Nombre	Tipo	Dirección
ARRAY_PRODUCTO_FINALIZADO_ENVIO	ARRAY[1..10] OF PRODUCTO	%MW230
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW230
ID	INT	%MW230
TIPO_PRODUCTO	INT	%MW231
BANDEJA	INT	%MW232
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW233
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW236
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW239
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW242
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW245
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW248
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW251
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW254
ARRAY_PRODUCTO_FINALIZADO_EN...	PRODUCTO	%MW257
ESTACION_1	BOOL	%MW260.0
ESTACION_2	BOOL	%MW260.1
ESTACION_3	BOOL	%MW260.2
CARGA_MATERIA_PRIMA_PT06	BOOL	%MW260.3
EXTRACCION_PRODUCTO_FINALIZADO	BOOL	%MW260.4
FLANCO_D_E_M_P	BOOL	%MW260.5
FLANCO_D_E_P_F	BOOL	%MW260.6
CONTADOR_ID	INT	%MW261
CONTADOR_P_FINALIZADO_ENVIO	INT	%MW262
CONTADOR_CANTIDAD_M_PRIMAS	INT	%MW263
CONTADOR_T_P_1	INT	%MW264
CONTADOR_T_P_2	INT	%MW265
CONTADOR_T_P_3	INT	%MW266
CONTADOR_E_1	INT	%MW267
CONTADOR_E_2	INT	%MW268
CONTADOR_E_3	INT	%MW269
ARRAY_TIEMPO_M_P	ARRAY[1..4] OF TIPO_TIEMPO	%MW270
ARRAY_TIEMPO_M_P[1]	TIPO_TIEMPO	%MW270
CRONO	TIME	%MW270
ID	INT	%MW272
ARRAY_TIEMPO_M_P[2]	TIPO_TIEMPO	%MW274
ARRAY_TIEMPO_M_P[3]	TIPO_TIEMPO	%MW278
ARRAY_TIEMPO_M_P[4]	TIPO_TIEMPO	%MW282
ARRAY_TIEMPO_P_F	ARRAY[1..4] OF TIPO_TIEMPO	%MW286
ARRAY_TIEMPO_P_F[1]	TIPO_TIEMPO	%MW286
CRONO	TIME	%MW286
ID	INT	%MW288
ARRAY_TIEMPO_P_F[2]	TIPO_TIEMPO	%MW290
ARRAY_TIEMPO_P_F[3]	TIPO_TIEMPO	%MW294
ARRAY_TIEMPO_P_F[4]	TIPO_TIEMPO	%MW298

Figura 154 - Variables enviadas por el PLC CAN

La primera función en el nodo function es la siguiente:

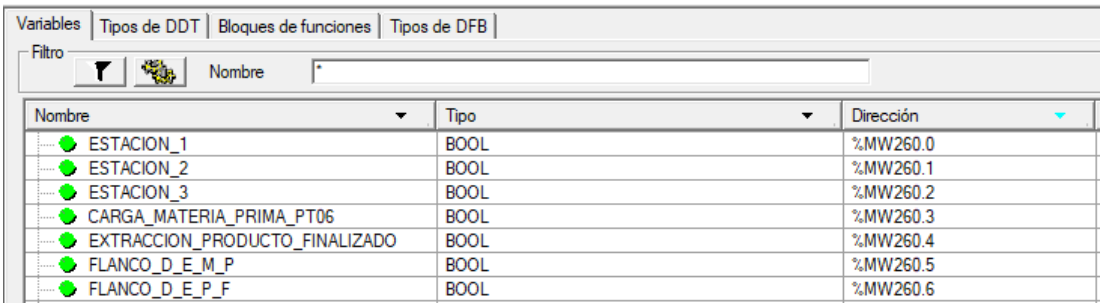
```
var coil = []

for(var j=0;j<7;j++){
        if((mbd[30]&Math.pow(2,j))===0)
    {
        coil[j] = 0;
    }
    else if((mbd[30]&Math.pow(2,j))>0)
    {
        coil[j] = 1;
    }

}
```

Figura 155 - Function leer booleanos %MW260

Esta función nos permite leer los valores booleanos guardados en la posición de memoria %MW260. En este espacio de memoria nos interesa conocer las posiciones de los 7 primeros bits (0 ->6). Este factor determina el valor de la variable "j". Si el valor es "1" la variable estará activada y si es "0" está desactivada. Por ejemplo, si la lectura de los 7 primeros bits de esta memoria da un valor 0000000 no habrá ninguna variable activada y si es 1111111 todas estarán activadas.



Nombre	Tipo	Dirección
ESTACION_1	BOOL	%MW260.0
ESTACION_2	BOOL	%MW260.1
ESTACION_3	BOOL	%MW260.2
CARGA_MATERIA_PRIMA_PT06	BOOL	%MW260.3
EXTRACCION_PRODUCTO_FINALIZADO	BOOL	%MW260.4
FLANCO_D_E_M_P	BOOL	%MW260.5
FLANCO_D_E_P_F	BOOL	%MW260.6

Figura 156 - Variables Booleanas envió CAN

Para conocer el estado de cada bit se realiza a partir de la multiplicación del estado de la memoria 260 por Math.pow. Math.pox devuelve la base elevada al exponente ($\text{base}^{\text{exponente}}$).

Por ejemplo:

7 primeros bits %MW260 0000000, Math.pow (2,0) = $2^{0} = 1 =$ 0000001 (binario)

%MW260	0000000
Math.pow	0000001
	0000000

7 primeros bits %MW260 0000001, Math.pow (2,0) =$2^{0} = 1 =$ 0000001 (binario)

%MW260	0000001
Math.pow	0000001
	0000001

7 primeros bits %MW260 0000001, Math.pow (2,1) = $2^{1} = 2 =$ 0000010 (binario)

%MW260	0000001
Math.pow	0000010
	0000000

7 primeros bits %MW260 0000010, Math.pow (2,1) = $2^{1} = 2 =$ 0000010 (binario)

%MW260	0000010
Math.pow	0000010
	0000010

El valor de cada bit se guarda en la fila (j) de la tabla coil[]. Si el resultado de la multiplicación da “0” (ejemplo: %MW260 0000000, j=0 y Math.pow (2,0) = 0000001) el valor guardado en la fila correspondiente al bit utilizado en Math.pow será “0”. En el ejemplo, se guarda en la fila 0, coil[0]=0. En cambio, si el valor es mayor que “0” se guardará el valor 1 en la fila correspondiente (por ejemplo: %MW260 0000010, j=1 y Math.pow (2,1) = 0000010). En este ejemplo, se guarda en la fila 1, coil[1]=1.

Las variables de la tabla coil que se utilizan para definir su valor son:

```
E_1 : coil[0],
E_2 : coil[1],
E_3 : coil[2],
E_M_P : coil[3],
EXTRACCION_P_FINAL = coil[4];
FLANCO_D_E_M_P = coil[5]
FLANCO_D_E_P_F = coil[6]
```

Figura 157 - Variables que utilizan la tabla coil

Para leer los valores de la tabla ARRAY_PRODUCTO_FINALIZADO_ENVIO (datos estación extracción producto finalizado) proporcionada por el PLC CAN (id, tipo producto y bandeja) se utiliza el siguiente código de programación:

```
EXTRACCION_P_FINAL = coil[4];
if(EXTRACCION_P_FINAL==1){
    if(i==31){
        i=1
    }
    for (var i=1;i<31;i=i+3){
        if (mbd[i] !== 0)
        {
        id = mbd[i-1]
        tipo_producto =  mbd[i]
        bandeja = mbd[i+1]
        }
    }
}

if (EXTRACCION_P_FINAL===0){
    id = 0
    tipo_producto =  0
    bandeja = 0
}
```

Figura 158 - Procesado Tabla producto finalizado

Cuando la estación está activa (EXTRACCION_P_FINAL = 1) se recorre todas las posiciones de la tabla, guardándose el valor del id, tipo producto y bandeja cuando el valor del id es diferente a “0”. Esto nos permite conocer cuál es la última fila de la tabla con su estructura rellena de datos y guardarlos en las variables de node-red.

Cuando la estación no está activa los valores de las variables id, tipo producto y bandeja tendrán valor “0” ya que no se está extrayendo ningún producto de esta estación.

Para leer los valores de la tabla ARRAY_TIEMPO_M_P (datos del cronómetro estación carga materias primas) proporcionada por el PLC CAN (id, crono) se utiliza el siguiente código de programación:

```
var id_crono_M_P_anterior = global.get("foo");
FLANCO_D_E_M_P = coil[5]

if (FLANCO_D_E_M_P===0){
    id_crono_M_P = 0
    crono_M_P =  0
}

if(FLANCO_D_E_M_P==1){
    if(x==54){
        x=40
    }
    for (var x=40;x<54;x=x+4){
        if (mbd[x+2] !== 0)
        {
        crono_M_P =  mbd[x]
        id_crono_M_P = mbd[x+2]
        }
    }
}
if (id_crono_M_P_anterior == id_crono_M_P){

    crono_M_P =  0
}

id_crono_M_P_anterior = id_crono_M_P
global.set("foo",id_crono_M_P_anterior);
```

Figura 159 - Procesado tabla tiempo estación materias primas

Cuando la estación no está activa los valores de las variables id y crono valdrán “0” ya que esta estación no está realizando ninguna actividad.

Cuando se está realizando alguna actividad (FLANCO_D_E_M_P=1) se recorre todas las posiciones de la tabla guardando el valor del id, crono cuando el valor del id es diferente a “0”. Esto nos permite conocer cuál es la última fila de la tabla con su estructura rellena de datos y guardarlos en las variables de node-red.

Utilizamos la variable global id_crono_M_P_anterior (útil en todos los nodos de Node-red) la cual tiene el valor del id anterior leído por node-red. Si este valor es igual al id leído en esta compilación el valor del crono será “0”. Esto nos permite enviar un solo valor de tiempo del crono por posición de la tabla.

Cronómetro de la estación de producto final (CAN), estación de trabajo 1 (Profibus), estación de trabajo 2 (Profibus) y estación de trabajo 3 (Profibus) siguen el mismo tipo de funcionamiento, solo varía la posición de la memoria de sus tablas y variables asociadas.

Para finalizar se crean tres measurements (aaa, bbb, ccc) con los siguientes datos:

```
msg.payload = [
 {
 measurement: "aaa",
 fields: {
E_1 : coil[0],
E_2 : coil[1],
E_3 : coil[2],
E_M_P : coil[3],
E_P_F : EXTRACCION_P_FINAL,
CAN_M_P : mbd[31],
CAN_P_F : mbd[32],
C_E_1 : mbd[37],
C_E_2 : mbd[38],
C_E_3 : mbd[39],
 }
 },
 {
 measurement: "bbb",
 fields: {
ID : id,
TIPO_PRODUCTO : tipo_producto,
BANDEJA : bandeja,
C_T_P_1 : mbd[34],
C_T_P_2 : mbd[35],
C_T_P_3 : mbd[36],
 }
 },
 {
 measurement: "ccc",
 fields: {
id_crono_M_P :  id_crono_M_P,
crono_M_P : crono_M_P,
id_crono_M_P_anterior : id_crono_M_P_anterior,
FLANCO_D_E_M_P : FLANCO_D_E_M_P,
m :m,
id_crono_P_F :  id_crono_P_F,
crono_P_F : crono_P_F,
id_crono_P_F_anterior : id_crono_P_F_anterior,
FLANCO_D_E_P_F : FLANCO_D_E_P_F,
 }
 }
];
return msg;
```

Figura 160 - Crear measurements aaa,bbb,ccc

Estos datos son recibidos por el nodo de Influx DB guardándolos en la base de datos.

El diagrama de flujo Node-red para recibir información del PLC de la línea Profibus.

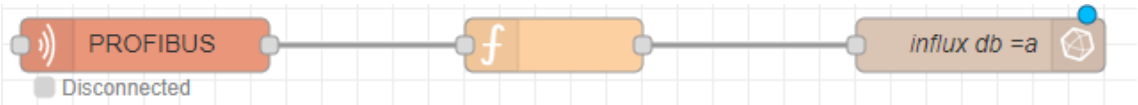


Figura 161 - Diagrama flujo Node red recibe información PLC Profibus

La primera función en el nodo function es la siguiente:

```
for(var a=0;a<3;a++){
        if((mbd[0]&Math.pow(2,a))===0)
    {
        coil_profibus[a] = 0;
    }
    else if((mbd[0]&Math.pow(2,a))>0)
    {
        coil_profibus[a] = 1;
    }

}
```

Figura 162 - Function leer booleanos

Se usó y funcionamiento es el mismo que el explicado anteriormente para los datos provenientes de la línea CAN. Se separan los valores de los 3 primeros bits de la posición de la memoria %MW199 y se guardan en la tabla coil_profibus[].

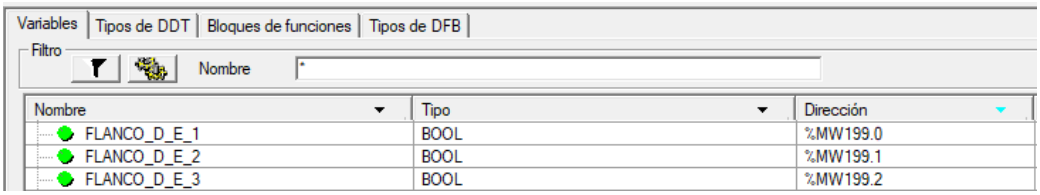


Figura 163- Variables Booleanas envió Profibus

Estas variables nos informan del estado de funcionamiento de las estaciones 1,2 y 3.

A continuación, se realiza la lectura de las tablas de tipo tiempo (información cronómetros estaciones) siguiendo el esquema de programación realizado en la línea Profibus para este tipo de tablas.

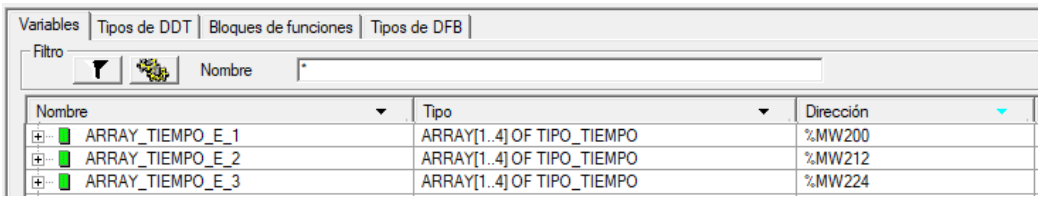


Figura 164 - Array tiempo Profibus

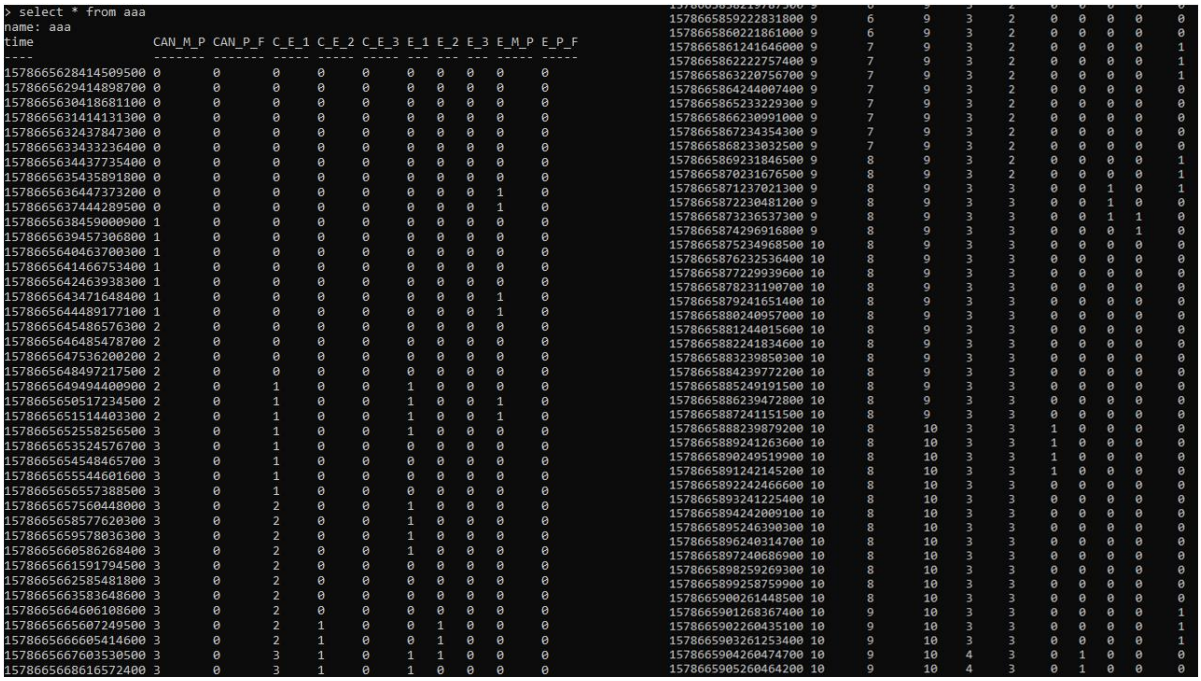
Se crea un 1 measurement (ddd) con los siguientes variables:

```
msg.payload = [
 {
 measurement: "ddd",
 fields: {
id_crono_E_1 :  id_crono_E_1,
crono_E_1 : crono_E_1,
id_crono_E_1_anterior : id_crono_E_1_anterior,
id_crono_E_2 :  id_crono_E_2,
crono_E_2 : crono_E_2,
id_crono_E_2_anterior : id_crono_E_2_anterior,
id_crono_E_3 :  id_crono_E_3,
crono_E_3 : crono_E_3,
id_crono_E_3_anterior : id_crono_E_3_anterior,
FLANCO_D_E_3 : FLANCO_D_E_3

 }
 }
 ];
return msg;
```

Figura 165 - Crear measurements ddd

Los valores de las variables son recibidos por el nodo de Influx DB guardándolos en la base de datos seleccionada y en los diferentes measurements con la estructura de las columnas elegida (fields) en el node red.



```
> select * from aaa
name: aaa
time                CAN_M_P CAN_P_F C_E_1 C_E_2 C_E_3 E_1 E_2 E_3 E_M_P E_P_F
----                ------- ------- ----- ----- ----- --- --- --- ----- -----
1578665628414509500 0       0       0     0     0     0   0   0   0     0
1578665629414898700 0       0       0     0     0     0   0   0   0     0
1578665630418681100 0       0       0     0     0     0   0   0   0     0
1578665631414131300 0       0       0     0     0     0   0   0   0     0
1578665632437847300 0       0       0     0     0     0   0   0   0     0
1578665633433236400 0       0       0     0     0     0   0   0   0     0
1578665634437735400 0       0       0     0     0     0   0   0   0     0
1578665635435891800 0       0       0     0     0     0   0   0   0     0
1578665636447373200 0       0       0     0     0     0   0   0   1     0
1578665637444289500 0       0       0     0     0     0   0   0   1     0
1578665638459000900 1       0       0     0     0     0   0   0   0     0
1578665639457306800 1       0       0     0     0     0   0   0   0     0
1578665640463700300 1       0       0     0     0     0   0   0   0     0
1578665641466753400 1       0       0     0     0     0   0   0   0     0
1578665642463938300 1       0       0     0     0     0   0   0   0     0
1578665643471648400 1       0       0     0     0     0   0   0   1     0
1578665644489177100 1       0       0     0     0     0   0   0   1     0
1578665645486576300 2       0       0     0     0     0   0   0   0     0
1578665646485478700 2       0       0     0     0     0   0   0   0     0
1578665647536200200 2       0       0     0     0     0   0   0   0     0
1578665648497217500 2       0       0     0     0     0   0   0   0     0
1578665649494400900 2       0       1     0     0     1   0   0   0     0
1578665650517234500 2       0       1     0     0     1   0   0   1     0
1578665651514403300 2       0       1     0     0     1   0   0   1     0
1578665652558256500 3       0       1     0     0     1   0   0   0     0
1578665653524576700 3       0       1     0     0     0   0   0   0     0
1578665654548465700 3       0       1     0     0     0   0   0   0     0
1578665655544601600 3       0       1     0     0     0   0   0   0     0
1578665656557388500 3       0       1     0     0     0   0   0   0     0
1578665657560448000 3       0       2     0     0     1   0   0   0     0
1578665658577620300 3       0       2     0     0     1   0   0   0     0
1578665659578036300 3       0       2     0     0     1   0   0   0     0
1578665660586268400 3       0       2     0     0     1   0   0   0     0
1578665661591794500 3       0       2     0     0     0   0   0   0     0
1578665662585481800 3       0       2     0     0     0   0   0   0     0
1578665663583648600 3       0       2     0     0     0   0   0   0     0
1578665664606108600 3       0       2     0     0     0   0   0   0     0
1578665665607249500 3       0       2     1     0     0   1   0   0     0
1578665666605414600 3       0       2     1     0     0   1   0   0     0
1578665667603530500 3       0       3     1     0     1   1   0   0     0
1578665668616572400 3       0       3     1     0     1   0   0   0     0
1578665859222831800 9       6       9     3     2     0   0   0   0     0
1578665860221861000 9       6       9     3     2     0   0   0   0     0
1578665861241646000 9       7       9     3     2     0   0   0   0     1
1578665862222757400 9       7       9     3     2     0   0   0   0     1
1578665863220756700 9       7       9     3     2     0   0   0   0     1
1578665864244007400 9       7       9     3     2     0   0   0   0     0
1578665865233229300 9       7       9     3     2     0   0   0   0     0
1578665866230991000 9       7       9     3     2     0   0   0   0     0
1578665867234354300 9       7       9     3     2     0   0   0   0     0
1578665868233032500 9       7       9     3     2     0   0   0   0     0
1578665869231846500 9       8       9     3     2     0   0   0   0     1
1578665870231676500 9       8       9     3     2     0   0   0   0     1
1578665871237021300 9       8       9     3     3     0   0   1   0     1
1578665872230481200 9       8       9     3     3     0   0   1   0     0
1578665873236537300 9       8       9     3     3     0   0   1   1     0
1578665874296916800 9       8       9     3     3     0   0   0   1     0
1578665875234968500 10      8       9     3     3     0   0   0   0     0
1578665876232536400 10      8       9     3     3     0   0   0   0     0
1578665877229939600 10      8       9     3     3     0   0   0   0     0
1578665878231190700 10      8       9     3     3     0   0   0   0     0
1578665879241651400 10      8       9     3     3     0   0   0   0     0
1578665880240957000 10      8       9     3     3     0   0   0   0     0
1578665881244015600 10      8       9     3     3     0   0   0   0     0
1578665882241834600 10      8       9     3     3     0   0   0   0     0
1578665883239850300 10      8       9     3     3     0   0   0   0     0
1578665884239772200 10      8       9     3     3     0   0   0   0     0
1578665885249191500 10      8       9     3     3     0   0   0   0     0
1578665886239472800 10      8       9     3     3     0   0   0   0     0
1578665887241151500 10      8       9     3     3     0   0   0   0     0
1578665888239879200 10      8       10    3     3     1   0   0   0     0
1578665889241263600 10      8       10    3     3     1   0   0   0     0
1578665890249519900 10      8       10    3     3     1   0   0   0     0
1578665891242145200 10      8       10    3     3     1   0   0   0     0
1578665892242466600 10      8       10    3     3     0   0   0   0     0
1578665893241225400 10      8       10    3     3     0   0   0   0     0
1578665894242009100 10      8       10    3     3     0   0   0   0     0
1578665895246390300 10      8       10    3     3     0   0   0   0     0
1578665896240314700 10      8       10    3     3     0   0   0   0     0
1578665897240686900 10      8       10    3     3     0   0   0   0     0
1578665898259269300 10      8       10    3     3     0   0   0   0     0
1578665899258759900 10      8       10    3     3     0   0   0   0     0
1578665900261448500 10      8       10    3     3     0   0   0   0     0
1578665901268367400 10      9       10    3     3     0   0   0   0     1
1578665902260435100 10      9       10    3     3     0   0   0   0     1
1578665903261253400 10      9       10    3     3     0   0   0   0     1
1578665904260474700 10      9       10    4     3     0   1   0   0     0
1578665905260464200 10      9       10    4     3     0   1   0   0     0
```

Figura 166 - Ejemplo base de datos

En el software Grafana se configura la base de datos añadiéndola. Se realiza en la menú configuración pulsando Add data source y especificando una base de datos tipo Influx.

Seleccionamos localhost y puerto 8086 ya que la base de datos se encuentra en el mismo ordenador de trabajo.

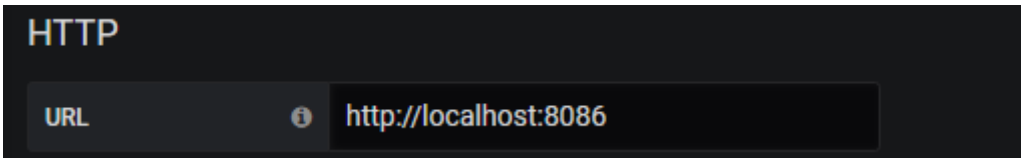


Figura 167 – HTTP URL

Seleccionamos la base de datos de lectura. Ejemplo: base de datos a.

Figura 168 InfluxDB Details Database

A continuación se crean los dashboard con sus widgets correspondientes.

En este proyecto se han creado 4 dashboard que muestran diferentes procesos de la línea.

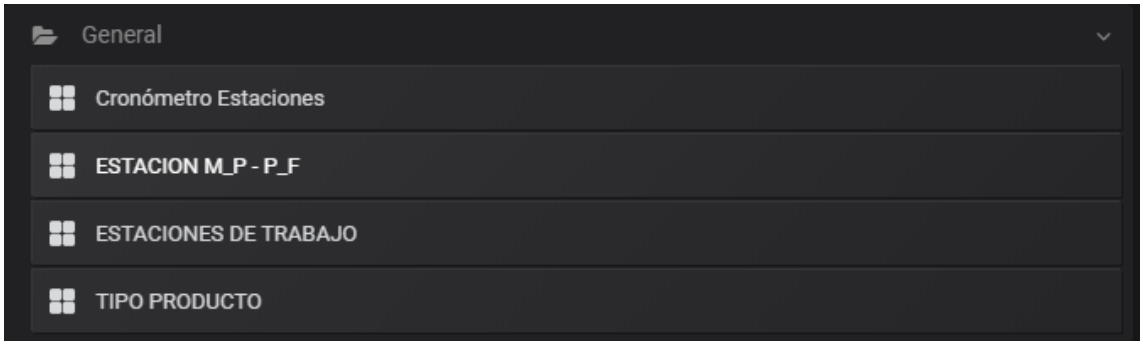


Figura 169 - Dashboard Grafana

A continuación se muestran la interacción de los dashboard del software Grafana con el flujo de producción siguiente:

Figura 170 - Flujo producción (Dashboard Node red)

En las estaciones de trabajo se observa el flujo de trabajo de las 3 estaciones a partir de unos gráficos y la cantidad total de piezas realizadas en cada estación.

Figura 171 - Dashboard Estaciones de trabajo

El flujo de producto finalizado se muestra en una gráfica en la que se ve el tipo de producto y en un panel de visualización la cantidad de producto extraído dependiendo del tipo de producto (1, 2, 3).

Figura 172 - Dashboard Tipo Producto

En estación M_P-P_F se observa el flujo de actividad de la estación de carga de materias primas y la estación extracción de producto finalizado gracias a unos gráficos. También se ve en un panel de visualización la cantidad total de materias primas cargadas y producto finalizado extraído.

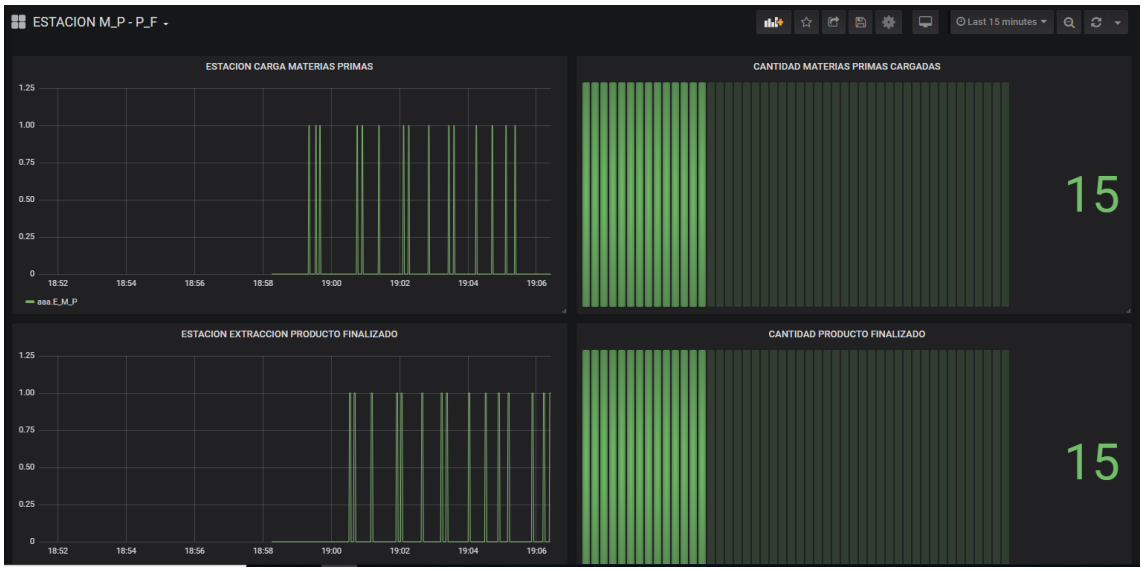


Figura 173 - Dashboard Estacion materias primas y producto final

En Cronómetros estaciones hay un gráfico por cada estación que nos muestra el tiempo de actividad en cada estación.

Figura 174 - Cronómetros estaciones

En el ejemplo anterior el tiempo en todas las estaciones es constante ya que el tiempo simulado siempre es el mismo. En cambio, en el siguiente ejemplo se ha variado el tiempo para cada entrada de producto.

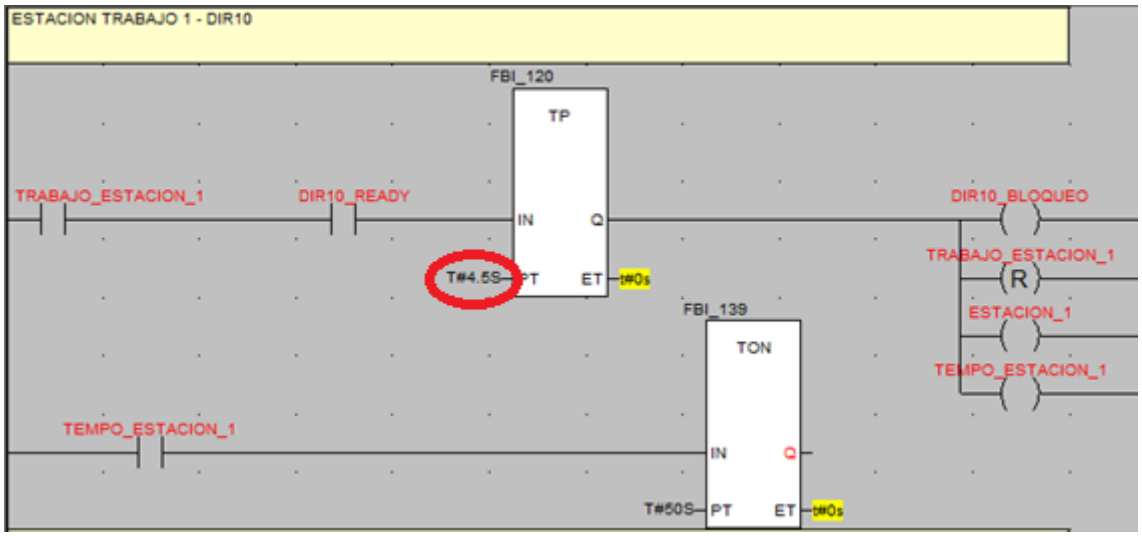


Figura 175 - Ladder cronometro estación de trabajo 1 cambio variable tiempo

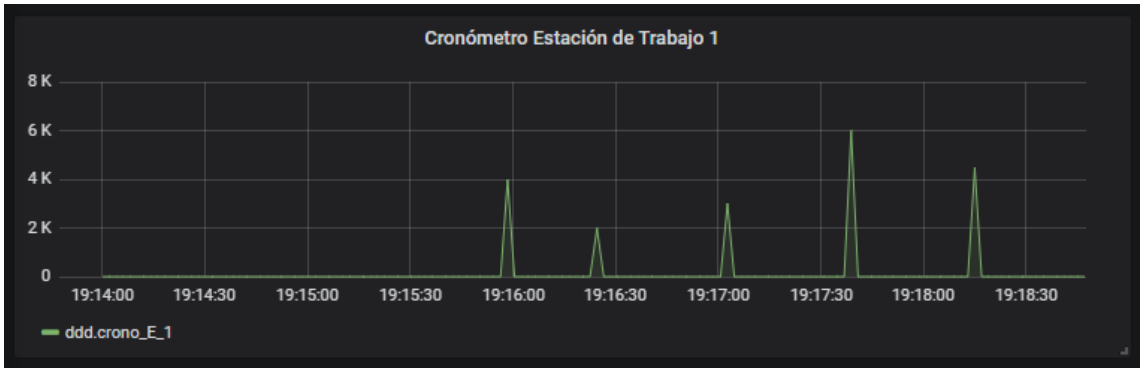


Figura 176 – Grafica cronometro estación de trabajo 1 cambio variable tiempo

5. Validación

Este apartado del proyecto consta de diferentes ensayos realizados durante la programación de la automatización y monitorización de la celda industrial con el objetivo de validar el sistema diseñado. Los ensayos se dividieron en pruebas funcionales y operativas. Las pruebas funcionales se realizaron para validar pequeñas configuraciones o bloques individuales y las operativas para validar todo el conjunto.

La primera validación fue la comprobación de las direcciones con los elementos de las líneas (plataformas, retenedores, motores etc.). El resultado esperado era que todas las direcciones estuvieran correctamente asociadas a su elemento. Se crearon o modificaron las direcciones que no estaban creadas o asociadas correctamente.

La segunda prueba fue la validación de una estación formada por un retenedor. Se realizó la prueba de funcionamiento y el resultado fue que el retenedor funcionaba correctamente.

La tercera prueba fue la validación de una estación formada por una plataforma. Se realizó la prueba de funcionamiento y en este caso el resultado no fue el esperado porque la subida de la plataforma se desactivaba al activarse el estado de reposo cuando el detector dejaba de detectar. La acción correctora fue la activación del reposo cuando el palet llegara a la estación destino.

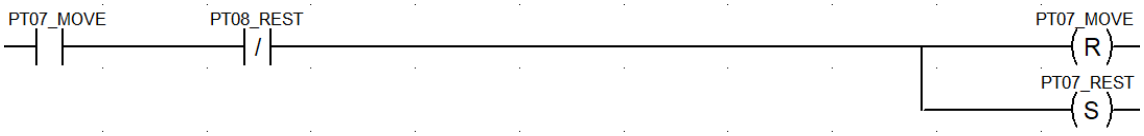


Figura 177 - Línea Ladder Move plataforma

La cuarta prueba fue la validación de la sección prioridades en la que se obtuvo el resultado esperado. Éste era que la línea funcionara cuando hubiera dos palets en espera para entrar en una intersección y que consecutivamente, entrara un palet de una estación diferente cada vez que se activara.

La quinta prueba fue la validación de las comunicaciones entre PLCs que permiten el intercambio de variables. El resultado obtenido fue satisfactorio después de buscar y recabar información sobre cómo realizarlo.

La sexta prueba fue la validación del funcionamiento de la elección de la dirección en las intersecciones. El resultado esperado era que al cambiar el estado de la dirección se activara o desactivara el MOVE con las otras puertas lógicas del avance permitiéndolo. El resultado fue satisfactorio.

La séptima prueba fue la validación de la programación de los retenedores, plataformas, intersecciones. El resultado obtenido no fue el esperado porque había estaciones que no funcionaban por fallos de escritura o asociación de variables. Al localizar los fallos y tomar la solución adecuada, la línea funcionaba correctamente pero apareció un fallo esporádico de aparición de bandejas fantasmas. Se solucionó agregando un temporizador TON para filtrar la señal. El orden de ejecución de los programas es la línea Ethernet siempre primero seguida de

la CAN o Profibus ya que la línea CAN necesita saber el estado del sensor DIR06 para realizar un funcionamiento correcto de la programación de esta estación.

La octava prueba fue la detección de los flancos ascendentes cuando se activaba una variable para la realización de los contadores. El resultado esperado en el PLC Profibus era su detección pero no lo realizaba correctamente. La acción de corrección fue la elaboración del siguiente código de programación que permite la detección de este flanco

```
(*PT09 ARRAY*)
DIR10_MOVE_E := DIR10_MOVE;
DIR10_MOVE_F_A := RE (DIR10_MOVE_E);
```

Figura 178 - Código ST detección flanco ascendente

La novena prueba fue la creación de las tablas de trazabilidad y sus contadores. El resultado esperado fue el trasvase correcto de información entre tablas. El resultado obtenido fue algún fallo en alguna tabla. Se reviso el código y se encontraron fallos de escritura o de asociación de variables al crearlo. Se arreglaron estos fallos buscándolos uno a uno y se validaron todas las tablas del programa hasta lograr que funcionaran correctamente. También se validó el funcionamiento de los resets de los contadores al llegar al máximo de longitud de las tablas. Una herramienta muy útil para comprobarlo fue mediante tablas de animación, como las tablas animadas utilizadas en la línea Profibus para comprobar su funcionamiento:

Nombre	Valor	Tipo
ARRAY_PT09_IN_PT08_OUT		ARRAY[1..10] O...
ARRAY_PT09_OUT_PT12_IN		ARRAY[1..10] O...
ARRAY_PT09_OUT_PT10_IN		ARRAY[1..10] O...
CONTADOR_PT09_IN_PT08_OUT		INT
CONTADOR_PT09_OUT_PT12_IN		INT
CONTADOR_PT09_OUT_PT10_IN		INT
ARRAY_PT10_OUT_PT14_IN		ARRAY[1..10] O...
ARRAY_PT10_OUT_PT12_IN		ARRAY[1..10] O...
CONTADOR_PT10_IN_PT09_OUT		INT
CONTADOR_PT10_OUT_PT14_IN		INT
CONTADOR_PT10_OUT_PT12_IN		INT
ARRAY_PT12_OUT_PT14_IN		ARRAY[1..10] O...
CONTADOR_PT12_IN_PT10_OUT		INT
CONTADOR_PT12_IN_PT09_OUT		INT
CONTADOR_PT12_OUT_PT14_IN		INT
ARRAY_PT14_OUT_PT16_IN		ARRAY[1..10] O...
CONTADOR_PT14_IN_PT10_OUT		INT
CONTADOR_PT14_IN_PT12_OUT		INT
CONTADOR_PT14_OUT_PT15_IN		INT

Figura 179 - Tablas de animación (tablas trazabilidad línea Profibus)

En las intersecciones donde entra un Palet y puede tomar dos direcciones no realizaba el correcto trasvase de información cuando el contador llegaba al máximo. La acción correctiva fue la creación de una variable de seguridad que permite realizar el reset en el momento deseado.

```
IF (DIR20_MOVE_F_A) = TRUE THEN

        INC (CONTADOR_PT16_IN_PT14_OUT);
        S_CONTADOR_PT16_IN_PT14_OUT := TRUE;
END_IF;

IF (PT16_MOVE_DIR07_F_A) = TRUE THEN

        INC (CONTADOR_PT16_OUT_PT04_IN);

        ARRAY_PT16_OUT_PT04_IN[CONTADOR_PT16_OUT_PT04_IN] := ARRAY_PT14_OUT_PT16_IN[CONTADOR_PT16_IN_PT14_OUT];
        S_CONTADOR_PT16_IN_PT14_OUT := FALSE;

END_IF;

IF (PT16_MOVE_PT17_F_A) = TRUE THEN

        (*CONTADORES PT16 ARRAY*)
        INC (CONTADOR_PT16_OUT_PT17_IN);

        ARRAY_PT16_OUT_PT17_IN[CONTADOR_PT16_OUT_PT17_IN] := ARRAY_PT14_OUT_PT16_IN[CONTADOR_PT16_IN_PT14_OUT];
        S_CONTADOR_PT16_IN_PT14_OUT := FALSE;

END_IF;
(*RESET CONTADOR*)

IF CONTADOR_PT16_IN_PT14_OUT = 10 AND S_CONTADOR_PT16_IN_PT14_OUT = FALSE THEN

        CONTADOR_PT16_IN_PT14_OUT := 0;

END_IF;
```

Figura 180 - Variable seguridad reset contadores

La décima prueba fue la elección de la dirección correcta en las intersecciones de la línea a partir de las tablas de trazabilidad. El resultado obtenido fue un fallo de elección en algunas intersecciones producido por errores a la hora de escribir el código de cada estación (fallos escritura o asociación de variables). Utilizando las tablas animadas de cada línea se encuentran los fallos y se van solucionando.

La décima primera prueba fue la validación del funcionamiento del dashboard de node red con el envío y recepción de datos. El resultado en casi todos los casos fue el esperado pero en algunos no se habían asociado correctamente las direcciones de las variables de los dos software. La acción correctora fue la revisión de las direcciones.

La decimosegunda prueba validaba el funcionamiento de las 5 estaciones de trabajo. El resultado obtenido fue el esperado aunque se solucionaron pequeños fallos producidos al crear el código.

En la estación formada por la plataforma PT05 el resultado esperado era que el palet vacío después de la extracción de producto entrara a la estación de carga de materias primas si era posible. El resultado obtenido fue que estos palets nunca entraban porque en la trazabilidad figuraba que había producto en ellos. Para que funcionara correctamente se procedió al borrado del id y tipo de producto una vez extraído el producto finalizado en la tabla ARRAY_PT04_OUT_PT05_IN.

```
ARRAY_PT04_OUT_PT05_IN[CONTADOR_PT04_OUT].ID := 0;
ARRAY_PT04_OUT_PT05_IN[CONTADOR_PT04_OUT].TIPO_PRODUCTO := 0;
```

Figura 181 - Borrado id y tipo producto tabla Pt04 Out pt05 In

La decimotercera prueba fue la validación del envió (PLC) y lectura/procesado (Node-red) de la información del último producto extraído en la línea guardada en la tabla ARRAY_PRODUCTO_FINALIZADO. Esta información era enviada y recibida correctamente hasta que todas las filas de la tabla de datos se llenaban. Esto sucedía debido a la programación de

node-red realizada ya que tiene en cuenta el valor del id para guardar las datos de la tabla y siempre guarda la posición de la tabla con el ultimo id diferente a 0. Esto provocaba que a partir del producto extraído con un valor superior a 11 el resultado siempre era el de la posición 10 en la tabla. La acción correctora fue crear una nueva tabla para el envío de los datos cuando el contador de producto finalizado era superior a 11, se iguala esta tabla a una tabla vacía borrando toda la información de datos almacenada hasta ese momento, permitiendo el procesado correcto por parte de node-red. No se eliminó la tabla ARRAY_PRODUCTO_FINALIZADO para no perder la información de esta estación. Por este motivo hay una tabla que es borrada y otra que sobrescribe información.

```
IF (CONTADOR_PRODUCTO_FINALIZADO = 11) THEN

ARRAY_PRODUCTO_FINALIZADO_ENVIO := ARRAY_VACIA;

CONTADOR_PRODUCTO_FINALIZADO := 1;

END_IF;
```

Figura 182 - Reset contador producto finalizado

La décimo cuarta prueba fue la validación del envió (PLC) y lectura/procesado (Node-red) de la información de las tablas donde se guardaba los cronómetros de las estaciones de trabajo. El resultado obtenido fue el funcionamiento incorrecto del código realizado en la función de node-red. La acción correctora fue la implementación de variables globales que nos permiten guardar valores de variable fuera del nodo o para su uso posterior en otra ejecución.

```
var id_crono_M_P_anterior = global.get("foo");
      global.set("foo",id_crono_M_P_anterior);
```

Figura 183 - Variable global Node red

Se tuvo en cuenta la acción correctora realizada en la validación anterior en las tablas utilizadas para almacenar la información de los cronómetros.

```
IF CONTADOR_TIEMPO_E_1 = 5 THEN
ARRAY_TIEMPO_E_1 := ARRAY_TIEMPO_VACIA;
CONTADOR_TIEMPO_E_1 := 1;
END_IF;
```

Figura 184 - Reset contador tiempo estacion 1

La decimoquinta prueba fue la validación de la base de datos (Influx DB) con los diferentes measurements y fields. El resultado obtenido fue el esperado ya que los datos suministrados por node-red se habían procesado correctamente.

La decimosexta prueba fue la validación del software Grafana. El resultado fue correcto al seguir las instrucciones de utilización y creación de dashboards.

6. Conclusiones

Una vez finalizado el proyecto se puede afirmar que su realización ha sido un éxito ya que se han podido implantar todos los objetivos definidos para la célula industrial.

El primer objetivo ha sido programar el funcionamiento de las estaciones de los retenedores o plataformas que me ha permitido utilizar y poner en práctica los conocimientos de lenguaje Ladder adquiridos en el grado de Ingeniería Electrónica Industrial y Automática.

El segundo objetivo ha consistido en la creación de la trazabilidad de la célula industrial y junto con el tercer objetivo en el que se ha realizado la simulación de estaciones de trabajo me ha ayudado a mejorar mis conocimientos en lenguaje estructurado (ST) y relacionarlo con la programación en los PLCs.

En el cuarto objetivo, la realización de un dashboard me ha permitido conocer el software Node-red y como enviar señales (variables del programa) a un PLC.

En cambio, el quinto objetivo he podido realizar lo contrario, enviar información del PLC a Node-red. En este objetivo se ha visualizado en un dashboard la información procedente de la línea. Esto me ha permitido familiarizarme con el software Influx DB para la creación de base de datos y el software Grafana para la creación de paneles de visualización en plataformas web.

Todos estos objetivos me han permitido interaccionar con los PLCs industriales ampliando y mejorando mis competencias en la elaboración de códigos de programación, aprender nuevos conceptos relacionados con los diferentes protocolos de comunicación y en el uso e implementación en la industria 4.0.

En relación a posibles trabajos en el futuro, a continuación añado varias propuestas para su aplicación al funcionamiento de la línea.

1. El uso de los estados de las estaciones y plataformas para realizar un panel de visualización de la línea que que indique el estado de cada una de las estaciones y si hay algún palet en movimiento entre dos estaciones.
2. La programación de la línea deber tener en cuenta paros de la línea como cambio de turnos, descansos, paradas para tareas de mantenimiento, etc.
3. También se podría Informar de los diferentes estados de la línea como producción, preparación, avería o mantenimiento.
4. Diseñar un dashboard en el que cada operario se tuviese que identificar para poder poner en funcionamiento la línea. Esto facilitaría la trazabilidad de los trabajadores para crear estudios de tiempo o cantidad de producción. Esto también permitiría tener un registro de producción para comprobar en los caso de devolución del producto por parte del cliente cuando se ha elaborado, turno, operario, incidencias durante la producción, etc.
5. Añadir una estación para realizar el control de calidad.
6. Por último se podría realizar el OEE (eficiencia general de los equipos) con los datos recabados de la célula industrial.

7. Bibliografía/web grafía

Manuales Schneider

https://www.se.com/ww/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/134000/FA134307/es_ES/Unity%20v50%20-%20Libreria%20Estandar.pdf
http://lra.unileon.es/sites/lra.unileon.es/files/Documents/plc/Unity_Pro/Manuales_Unity/Manual_Unity.pdf
http://lra.unileon.es/sites/lra.unileon.es/files/Documents/plc/Unity_Pro/Manuales_Unity/Unity_Manual%20de%20Referencia.pdf
https://www.se.com/ww/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/33000/FA33773/es_ES/Unity%20Pro%20Modalidades%20de%20Funcionamiento_33003104_K01_000_20.pdf
https://instrumentacionycontrol.net/wp-content/uploads/2017/11/IyCnet_GuiaRapidaUnityPRO-min.pdf

PLC

https://www.ingmecafenix.com/automatizacion/que-es-un-plc/
https://intrave.wordpress.com/2015/02/20/para-que-sirve-un-plc/
http://www.ctinmx.com/que-es-un-plc/
http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE6_1_1.pdf
https://es.wikipedia.org/wiki/Controlador_l%C3%B3gico_programable
https://comofunciona.co.com/un-plc/

Maestro/esclavo – cliente/servidor

https://prezi.com/iyg7zmnlztwk/estructura-interfaz-de-comunicacion-maestro-y-esclavo/
https://www.researchgate.net/figure/Figura-4-Modelo-Maestro-Esclavo-El-maestro-es-el-que-inicia-un-requerimiento-de_fig4_279531120
https://product-help.schneider-electric.com/ED/ES_Power/MTZ_Modbus_Guide/EDMS/DOCA0105ES/DOCA0105xx/Master_NTX_NWX_Modbus_Protocol/Master_NTX_NWX_Modbus_Protocol-2.htm
http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/cliente-servidor.html

Producto/consumidor

https://www.uv.es/rosado/courses/sid/Capitulo3_rev0.pdf
http://www.ni.com/tutorial/3023/es/
https://es.wikipedia.org/wiki/Problema_productor-consumidor

Modbus y Modbus TCP/IP

https://ricveal.com/blog/modbus/
https://ricveal.com/blog/modbus/
http://www.simplymodbus.ca/FAQ.htm
https://www.eeymuc.co/31-protocolo-modbus/
https://support.industry.siemens.com/tf/ww/en/posts/difference-between-profibus-and-modbus/170445?page=0&pageSize=10
https://www.automation.com/automation-news/article/profibus-and-modbus-a-comparison
https://es.wikipedia.org/wiki/Suma_de_verificaci%C3%B3n
https://www.logicbus.com.mx/Modbus.php
http://automation-networks.es/glossary/modbus-tcpip
https://cdn.automationdirect.com/static/manuals/hxecommsp/ch5.pdf

https://www.logicbus.com.mx/Modbus.php
http://automation-networks.es/glossary/modbus-tcpip
http://bibing.us.es/proyectos/abreproy/90946/fichero/marhertin.pdf
https://www.logicbus.com.mx/blog/modbus-tcp-ip/
https://www.eeymuc.co/31-protocolo-modbus/
https://ricveal.com/blog/modbus/
https://www.ingmecafenix.com/automatizacion/sensor-proximidad-capacitivo/
https://www.ni.com/es-es/innovations/white-papers/14/the-modbus-protocol-in-depth.html

Profibus

https://es.scribd.com/document/149687428/Que-es-PROFIBUS
http://josemiron.blogspot.com/2015/09/que-es-profibus.html
http://www.etitudela.com/entrenadorcomunicaciones/downloads/profibusfuncionamientoteorico.pdf
https://www.uv.es/rosado/courses/sid/Tema3_Profibus_transp.pdf
http://www.autracen.com/profibus-vs-profinet-vs-profisafe/
https://www.uv.es/rosado/courses/sid/Tema3_Profibus_transp.pdf
https://upcommons.upc.edu/bitstream/handle/2099.1/4298/Profibus.pdf?sequence=10&isAllowed=y

CAN

http://oa.upm.es/48054/8/TFM_ADRIAN_MARTINEZ_REQUENA.pdf
https://petrolheadgarage.com/Posts/caracteristicas-de-un-sistema-can-bus/
https://www.youtube.com/watch?time_continue=104&v=wWTNe3o_Whg&feature=emb_title
https://www.ni.com/es-es/innovations/white-papers/06/controller-area-network--can--overview.html
https://es.wikipedia.org/wiki/Bus_CAN
https://www.motorpasion.com/coches-hibridos-alternativos/can-bus-como-gestionar-toda-la-electronica-del-automovil
https://gpstotal.org/es/sensor-automotriz/que-es-canbus
https://www.uv.es/rosado/courses/sid/Capitulo3_rev0.pdf
http://gro.usal.es/trabajos/Proyecto%20Raquel/web/index_archivos/CANopen.pdf
https://es.scribd.com/document/311673866/PROTOCOLO-CANOPEN
https://www.festo.com/cms/es_es/16368.htm
https://es.wikipedia.org/wiki/Modelo_OSI#Capa_de_aplicaci%C3%B3n_-_Capa_7

Ethernet

https://es.wikipedia.org/wiki/Ethernet
https://www.linksys.com/es/r/resource-center/que-es-ethernet/
https://tecnologia-facil.com/que-es/que-es-ethernet/
https://www.ecured.cu/Ethernet
https://www.speedcheck.org/es/wiki/ethernet/

HTTP

https://developer.mozilla.org/es/docs/Web/HTTP
http://informaikta.blogspot.com/p/hipermedia.html
https://concepto.de/http/

Elementos de la celda

https://www.keyence.com.mx/ss/products/sensor/sensorbasics/photoelectric/info/
http://dominion.com.mx/descargas/sensores-fotoelectricos.pdf
https://fidestec.com/blog/sensores-fotoelectricos-industriales-fotocelulas/
https://es.wikipedia.org/wiki/Sensor_capacitivo

Pasarela

https://www.incibe-cert.es/blog/evolucionando-comunicacion-industria

Node-red

https://www.toptal.com/nodejs/programacion-visual-con-node-red-conectando-el-internet-de-las-cosas-con-facilidad
https://en.wikipedia.org/wiki/Node-RED
https://aprendiendoarduino.wordpress.com/2018/11/20/node-red/
https://ricveal.com/blog/node-red-construye-el-internet-de-las-cosas/
https://about.sofia2.com/2016/11/16/que-es-nodered/
https://stackoverflow.com/questions/44298263/is-it-possible-to-declare-global-variables-in-node-red-and-use-them-on-the-node
https://www.youtube.com/watch?v=e70ta8jI_nM
https://www.youtube.com/watch?v=P6MyTXEb71I
https://www.youtube.com/channel/UCftGMspZELFu01LQVtJPGtg

https://material.io/resources/icons/?style=baseline
https://flows.nodered.org/node/node-red-contrib-modbustcp

Influxdb

https://robertoorayen.eu/2016/11/21/conceptos-basicos-influxdb/
http://support.ptc.com/help/thingworx_hc/thingworx_8_hc/es/index.html#page/ThingWorx/Help/Composer/DataStorage/PersistenceProviders/using_influxdb_as_the_persistence_provider.html
https://thecocktail.engineering/stack-de-monitorizacion-con-kubernetes-a2bc8e556ed8
https://en.wikipedia.org/wiki/InfluxDB
https://www.jorgedelacruz.es/2016/06/29/en-busca-del-dashboard-perfecto-influxdb-telegraf-y-grafana-parte/

Grafana

https://www.muutech.com/tecnologias/grafana-con-demo-online/
https://es.wikipedia.org/wiki/Grafana
https://pandorafms.com/blog/es/que-es-grafana/
https://www.linuxito.com/cloud/1104-creando-mi-primer-dashboard-en-grafana